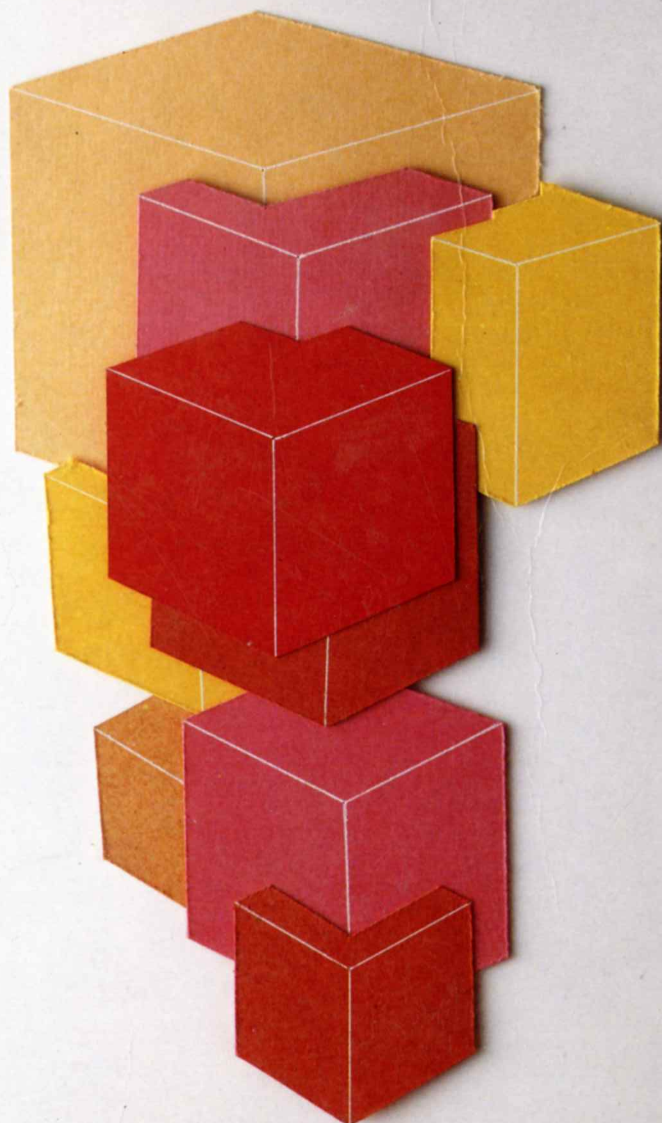
BASES DE DATOS RELACIONALES

E. RIVERO CORNELIO



PARANINFO S.A.

bases de datos relacionales

bases de datos relacionales

E. RIVERO CORNELIO

1988

PARANINFO S.A.

MADRID

Impreso en España
Printed in Spain

ISBN: 84-283-1652-X

Depósito legal: M. 34.600.—1988



Magallanes, 25 - 28015 MADRID (09225/38/10)

Artes Gráficas BENZAL, S.A., Virtudes, 7 - 28010 MADRID

Dedicatoria

A Reme.

Indice de materias

Prólogo

Dentro de la tecnología informática, las técnicas de bases de datos han ido aumentando su importancia y sus aplicaciones en los últimos años. La razón para ello ha estado en la demanda, siempre creciente, de utilización de grandes masas de datos, frecuentemente integrados, y con requisitos de alta productividad en su manejo, tanto en el desarrollo de aplicaciones por programadores profesionales, como por parte de usuarios finales. Las bases de datos han respondido a esta necesidad, permitiendo estructurar éstos con técnicas más formalizadas, de uso más sencillo y con mayor capacidad para reflejar su significado.

En la búsqueda de estos objetivos, las bases de datos relacionales han supuesto un avance fundamental. Sus principios teóricos se establecieron y consolidaron en la década de los 70, y en la actual, la de los 80, están pasando del ámbito de la investigación al de las realizaciones prácticas, expandiéndose rápidamente sus aplicaciones. Al mismo tiempo, continúa la investigación teórica, pues los conceptos relacionales forman una base sólida para la exploración de nuevas áreas, como por ejemplo el enriquecimiento semántico del modelo de datos, las bases de datos distribuidas, nuevos tipos de datos (gráficos, voz), uso de técnicas de inteligencia artificial para formar bases de conocimiento y otros.

El primer paso para utilizar una base de datos relacional es hacer un buen diseño de sus estructuras de datos. A ello va encaminado este libro, que, por tanto, no contempla otros aspectos como concurrencia de acceso, protección de la confidencialidad de datos, actualización de vistas, optimización de consultas, recuperación de datos, etc., temas todos ellos muy importantes para otros fines. El libro va orientado a conseguir diseñar adecuadamente estructuras relacionales de datos que sean coherentes con el significado de éstos.

Para ello, el plan seguido es el siguiente:

1) Exponer las definiciones básicas del modelo relacional de datos, así como el lenguaje llamado álgebra relacional, que es necesario para posteriormente tratar la normalización de relaciones, basada en la utilización de operaciones algebraicas (fundamentalmente proyección y yunción). Esto se expone en los dos primeros capítulos.

2) Aunque no estrictamente necesario para el objetivo del libro, se incluye una descripción del lenguaje llamado cálculo relacional, por ser éste la base de otros que se usan en la práctica. Esto se encuentra en el tercer capítulo.

3) Exponer los conceptos y teoría de normalización de relaciones (capítulo cuarto), fundamentales para comprender cómo conseguir un buen diseño. Sobre este tema existe una abundante bibliografía que, frecuentemente, o bien lo expone con una orientación muy matemática, apta sólo para especialistas, o bien lo presenta sin el rigor necesario al tratar de vulgarizarlo excesivamente. Este libro pretende no caer en ninguno de estos extremos.

4) En los dos últimos capítulos se propone un método de diseño basado, con algunas modificaciones, en los conceptos de entidad y relaciones. Se da una definición de entidad más formalizada de la que suele manejarse habitualmente, apoyada en los tipos de asociaciones, que no contradice ni elimina el uso intuitivo de este concepto, sino que más bien lo refuerza haciéndolo más preciso. Este método se aplica a varios casos prácticos.

Al final de cada capítulo, excepto el primero, se proponen algunos ejercicios cuyas soluciones se muestran en el apéndice. El lector podrá ejercitar y aclarar en ellos algunos de los conceptos expuestos, que además se ha intentado que vayan acompañados en el texto por numerosos ejemplos. Estos últimos, junto con los ejercicios al final de cada capítulo y los casos prácticos del último, pretenden también desarrollar en el lector una percepción de lo complejo, esquivo y multiforme que es el aspecto semántico de los datos. Para el lector que desee profundizar en esto, es recomendable la lectura del libro del señor Kent *Data and Reality*, citado en la bibliografía.

En las definiciones ha sido inevitable usar términos del lenguaje matemático para dar precisión a los conceptos, pero se ha intentado siempre reducir el formalismo matemático a lo imprescindible, dando prioridad a la claridad y comprensión de los conceptos antes que al aparato que los soporta. Por ello no se incluyen, en general, demostraciones de las propiedades que se enuncian, salvo en algunas pocas ocasiones en que el resultado que se demuestra es muy importante para el desarrollo posterior, o a modo ilustrativo del tipo de demostraciones que suele haber en la literatura especializada. El lector que desee profundizar en este aspecto puede recurrir a algunos de los libros citados en la bibliografía. En resumen, se ha pretendido que no sea necesario un previo conocimiento matemático especializado para la lectura y comprensión de esta obra, que aspira a ser útil tanto a los profesionales de proceso de datos como a los estudiantes interesados por estas materias.

El autor, que ejerce su actividad profesional en IBM S.A.E., desea aclarar que todos los conceptos y puntos de vista expuestos son de su exclusiva responsabilidad y no reflejan necesariamente la posición de IBM sobre estos temas.

También desea agradecer el ánimo y colaboración que le han prestado algunos colegas y compañeros, especialmente el señor J. J. Kuntz.

Y finalmente, siendo consciente de que esta obra es mejorable, pide la benevolencia del lector para las carencias e incorrecciones que pueda tener, y sólo le queda desearle esperanzadamente una lectura interesante, instructiva y provechosa.

Definiciones y conceptos básicos

1

INFORMACION Y DATOS

Información

Este libro trata, en términos generales, de cómo representar la información mediante datos. Dado que tanto el concepto de información como el de datos se utilizan en muy diferentes contextos y con múltiples matices, vamos a intentar aquí acotar el significado de estas palabras en el ámbito que nos interesa.

La información puede adoptar formas muy diversas. Por ejemplo:

- *Signos escritos:* Libros, periódicos, facturas, letras de cambio, apuntes de contabilidad, cartas, jeroglíficos, dibujos, partituras musicales, inscripciones en piedra, en papiro, etc.
- *Sonidos:* Lenguaje hablado, música, los silbidos que usan en la isla de la Gomera, etc.
- *Señales de cualquier tipo:* Las señales de tráfico, los lenguajes con banderas entre los barcos, las señales de humo entre los indios del lejano Oeste, los garbanzos que Pulgarcito arrojaba para marcar el camino, etc.

En todos estos ejemplos hay algo común: un mensaje. Es decir, unas ideas vertidas por un ser humano sobre un soporte material para que otro ser humano las reciba. Las ideas representadas por el mensaje son la información de éste. Para nosotros, por tanto, el concepto de información va unido al proceso de la comunicación humana.

Así, no estamos aquí interesados en las ideas que puedan surgir en la mente de un observador al recibir estímulos procedentes de fenómenos naturales, aunque podrían considerarse información en un sentido amplio. No estamos, por ejemplo, interesados en los mensajes de comunicación entre animales, que pueden en cambio ser objeto de estudio y aporte de información para un biólogo o un cazador. Tampoco consideraremos como información los sentimientos que una hermosa puesta de sol pueda provocar en una persona sensible.

En cambio, sí entrarían en el concepto de información enunciado los sentimientos provocados en una persona al contemplar un cuadro que represente una puesta de sol. Aquí existe un proceso de comunicación humana. El cuadro es un mensaje, fruto consciente de una mente humana que desea comunicar parte de su mundo interior a otras, y aporta una información, que es el conjunto de emociones y sentimientos que surgen en la mente de quien lo contempla.

Nuestro ámbito de estudio es más restringido. Sólo nos interesa la información que pueda representarse en mensajes procesables por máquinas. Los ordenadores son máquinas que procesan cadenas de signos o caracteres, a gran velocidad y en grandes volúmenes.

Por tanto, nos restringiremos a la información aportada por mensajes formados por cadenas de signos.

Datos

Analicemos qué tipos de mensajes son los que nos interesan.

Consideraremos mensajes construidos con signos, combinando éstos en unidades de complejidad creciente ateniéndose a unas reglas que vienen definidas por el lenguaje en que se escribe el mensaje.

Para construir estos mensajes partimos de un conjunto finito que llamaremos alfabeto. A los elementos de este conjunto los llamaremos signos.

Concatenando signos del alfabeto se forman unas cadenas, de longitud finita. No todas las combinaciones posibles de signos forman cadenas válidas en un lenguaje determinado. A las que lo son, las llamaremos palabras. El conjunto de todas las palabras forma el vocabulario del lenguaje. (Esta definición de palabra sería demasiado general para estudiar un idioma, pues incluiría como palabras a los signos de puntuación, pero es suficiente para nuestro propósito.)

Un mensaje se construye concatenando un número finito de palabras, según unas reglas definidas en la gramática del lenguaje, y aporta al que lo recibe una información constituida por todas las ideas, emociones, sentimientos, etc., que aparecen en su mente como consecuencia de recibirlo. Llamaremos significado de un mensaje a la información que éste aporta a la persona que lo recibe. De acuerdo con esto, el significado de un mensaje es algo subjetivo, que depende de la persona receptora. Así, por ejemplo, un libro no tiene apenas significado para un analfabeto, puesto que a su mente no llega el mensaje que contiene.

En principio, el lenguaje en que se escribe un mensaje puede ser cualquiera. Si se utiliza un lenguaje humano de uso común, por ejemplo el castellano, el francés o el latín (en su día fue el más común), diremos que el mensaje no tiene formato, o que está escrito en texto libre. Este tipo de mensajes y la información por ellos representada es objeto de numerosas aplicaciones de los ordenadores en las áreas de proceso y edición de textos, tratamiento de documentos y automatización de oficinas. Pero no, tampoco es el objeto de nuestro estudio.

Muy frecuentemente las aplicaciones de los ordenadores procesan mensajes construidos con un lenguaje muy simple, en el que normalmente cualquier

combinación obtenida concatenando palabras del vocabulario constituye un mensaje formalmente válido. Cada palabra de un mensaje tiene un significado predefinido en función de la posición que ocupa dentro de él. Las palabras que forman el vocabulario se llaman *datos*.

Usando una terminología más común en informática, cada mensaje es un registro y cada registro contiene varias palabras cuyo significado está ligado a su posición dentro del registro. El «hueco» o posición que dentro de cada registro ocupa un dato se llama campo. Los registros que tienen campos con el mismo significado predefinido se suelen agrupar en conjuntos llamados ficheros o archivos.

Los mensajes que nos interesan son un subtipo de estos registros, formado por aquellos que tienen un número fijo de campos. En este caso los conjuntos de registros de igual significado, y por consiguiente con el mismo número de campos, se llaman tablas o relaciones. Los campos se suelen llamar atributos.

Este tipo de registros, sus conjuntos o tablas, sus componentes, atributos y datos, y la información representada por ellos, constituyen finalmente la materia que vamos a explorar en las páginas siguientes.

ESQUEMA CONCEPTUAL DE DATOS

Sea una empresa o entidad de cualquier tipo que desea almacenar datos que reflejan información sobre sus actividades. En la figura siguiente se representan los pasos para conseguirlo.

Hay que empezar acotando qué parcela del mundo exterior nos interesa representar en los datos. Estará formada por todos los objetos y acontecimientos cuyo conocimiento nos permita una mejor gestión de nuestras actividades.

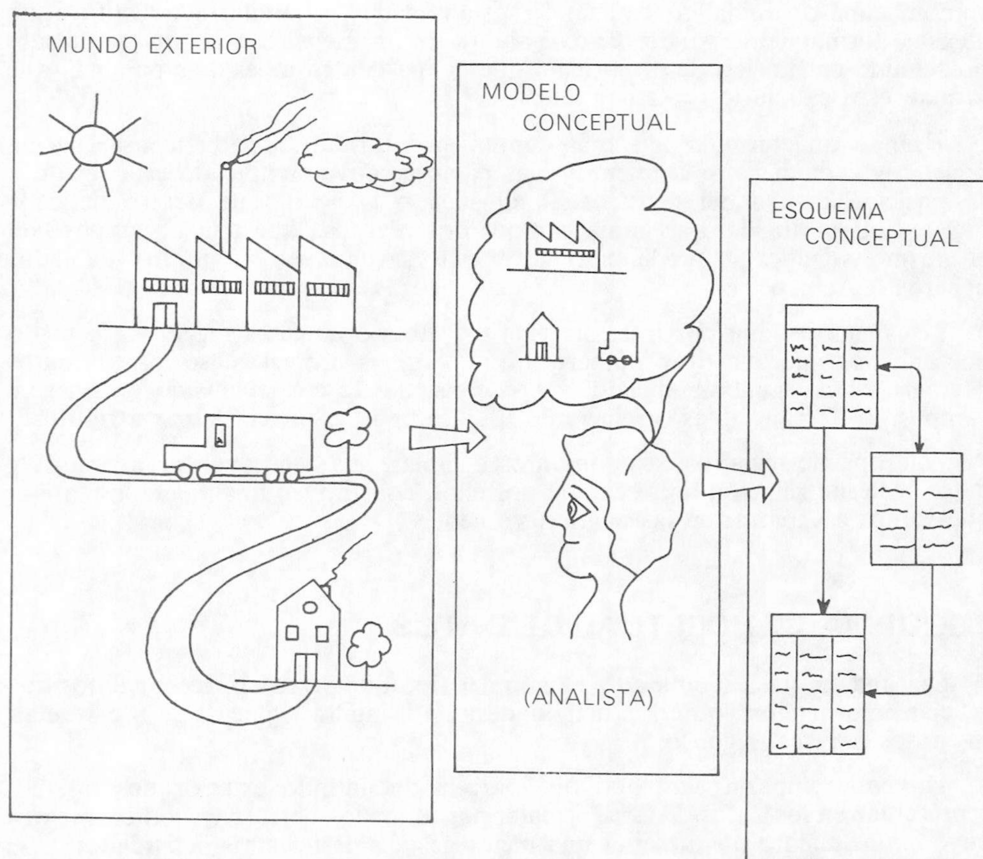
El diseñador, o analista de datos, debe aprehender, comprender y conceptualizar este mundo, transformándolo en un conjunto de ideas y definiciones, que formen una imagen fiel del comportamiento del mundo real. A esta imagen del mundo exterior la llamaremos modelo conceptual. Para construir un buen modelo, el analista debe utilizar una gran dosis de procesos mentales de abstracción, análisis y síntesis. Necesitará además la colaboración de personal directivo.

Una vez definido el modelo conceptual, el analista lo transforma en una descripción de datos, atributos y tablas, incluyendo las posibles interrelaciones entre estos elementos y su significado. A esta descripción la llamaremos esquema conceptual de datos.

A la operación de transformar el modelo conceptual en un esquema conceptual la llamaremos diseño lógico de datos (por ello, al esquema conceptual también lo llamaremos a veces esquema de diseño).

Una vez definido el esquema conceptual, hay que traducirlo a estructuras almacenables en soportes físicos controlados por el ordenador, normalmente discos magnéticos. Esta transformación se suele llamar diseño físico de datos.

1



Un buen diseño lógico debe producir un esquema conceptual que sea una imagen fiel y completa del modelo conceptual, incluyendo algunas interrelaciones y condiciones semánticas, es decir, aquellas que son consecuencia del significado de los datos.

Por ello vamos a explorar en los siguientes capítulos algunos tipos de condiciones semánticas y cómo expresarlas en el esquema conceptual. En los últimos capítulos propondremos un método y una serie de reglas que sirven de ayuda al analista en el proceso de diseño lógico.

Empezaremos, en los apartados siguientes, definiendo los objetos con los que se construye el esquema conceptual: atributos, tablas, dominios, etc.

DOMINIOS

Un dominio, Di, es un conjunto de palabras, es decir, de ristras de caracteres tomados de un alfabeto dado. En general nos interesará definir también un

conjunto de operadores sobre estas palabras, para poder manipularlas. Por ejemplo, operaciones aritméticas y de comparación.

Un dominio puede ser un conjunto finito o infinito:

Ejemplos:

- Alfabeto: (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, /, –, ').
- Dominio D_1: (0, 10, 11, 12, 13, 14, 15).
- Dominio D_2: Conjunto de los números naturales.
- Dominio D_3: (0, 1, 2, 1/2, 1/3, 2/3).
- Dominio D_4: Conjunto de los números racionales.
- Dominio D_5: (0, 1, 2, –1, –2).
- Dominio D_6: (0, 3,14, –3,14).

D_2 y D_4 son infinitos.

El producto cartesiano de dos conjuntos, D_i y D_j, es otro conjunto, cuyos elementos son todas las parejas de valores que se pueden formar tomando un elemento de D_i y otro de D_j. Lo representamos como $(D_i \times D_j)$.

Ejemplo:

$(D_5 \times D_6)$ es el conjunto: (⟨0, 0⟩, ⟨0, 3,14⟩, ⟨0, –3,14⟩, ⟨1, 0⟩, ⟨1, 3,14⟩, ⟨1, –3,14⟩, ⟨2, 0⟩, ⟨2, 3,14⟩, ⟨2, –3,14⟩, ⟨–1, 0⟩, ⟨–1, 3,14⟩, ⟨–1, –3,14⟩, ⟨–2, 0⟩, ⟨–2, 3,14⟩, ⟨–2, –3,14⟩).

RELACIONES

Supongamos un conjunto, **D**, de dominios:

$\mathbf{D} = (D_1, D_2, D_3, \ldots, D_n)$.

Definamos entre ellos un criterio de orden que nos permita colocarlos a todos en una secuencia de menor a mayor (o de precedente a siguiente):

Secuencia: $D_1 < D_2 < D_3 < \cdots \langle D_n$.

Formemos ahora un producto cartesiano con algunos o todos los dominios de **D**:

$D_1 \times D_2 \times \cdots \times D_k$.

Los factores del producto anterior pueden estar repetidos. Por tanto, puede ser $k = n$, $k < n$ o $k > n$.

Los factores los colocamos en el orden definido por la secuencia citada anteriormente.

Ejemplo:

$D_1 \times D_2 \times D_3$; $D_1 \times D_1 \times D_4$; $D_1 \times D_2 \times D_2 \times D_3$; etc.

Una relación, R, es un subconjunto finito de un producto cartesiano formado con las reglas expuestas. Si representamos con $\subseteq$ la propiedad de que un conjunto esté contenido en otro tendríamos:

$R \subseteq (D_1 \times D_2 \times D_3 \times \cdots \times D_k)$.

Cada elemento de una relación se llama tupla.

Se llama grado de una tupla al número de componentes que tiene. El grado de una relación es el de sus tuplas.

Como R es un conjunto, todas sus tuplas deben ser diferentes. Es decir, en una relación no puede haber tuplas repetidas. Además, no hay un criterio de orden definido entre ellas.

ATRIBUTOS

A cada dominio que participa en una relación se le da un nombre que lo identifica de forma única para esa relación.

Entonces una tupla puede representarse o bien como una lista ordenada de valores:

$t = \langle t_1, t_2, t_3, \ldots, t_k \rangle$, donde:

t_1 = Valor del primer dominio,
t_2 = Valor del segundo dominio,
t_3 = Valor del tercer dominio, etc.

O bien como una lista desordenada de valores identificados por sus nombres de atributos:

$t = \langle ATR3 = t_3, ATR2 = t_2, ATR1 = t_1, \ldots, ATRk = t_k \rangle$.

CLAVES

Ya hemos dicho que como una relación R es un conjunto, todas sus tuplas han de ser diferentes, es decir, no se repiten sus valores.

A todo conjunto de atributos con esta propiedad se le llama superclave. Sea una superclave, tal que la formada por los atributos A, B, C y D de una relación R. Si no es posible quitarle ninguno de sus atributos sin que pierda su cualidad de superclave, se dice que es una clave de R. En definitiva, una clave de R es un conjunto de atributos de R que no puede tomar valores repetidos, y tal que

cualquier subconjunto suyo sí puede tomar valores repetidos, es decir, no es una superclave.

Toda relación tiene al menos una clave. Pero puede tener más de una. Esto depende del significado de la relación.

Entre los valores que toma una clave de una relación y las tuplas de ésta existe una correspondencia biunívoca, de modo que las claves pueden utilizarse como identificaciones de las tuplas de la relación. El único medio de dirigirnos a una tupla determinada, distinguiéndola de todas las demás, es mediante los valores de los atributos de alguna de sus claves.

TABLAS

Si colocamos todas las tuplas de una relación una debajo de otra, alineando los componentes correspondientes de cada atributo en una columna, y colocamos en la cabecera de cada columna el nombre de su atributo, obtenemos una representación de la relación a la que designaremos con el nombre de TABLA. Cada tupla forma una fila de la tabla.

EXTENSION E INTENSION DE UNA RELACION

Al transcurrir el tiempo, una relación puede sufrir modificaciones para reflejar los cambios del mundo real que representa.

Al añadir, borrar o modificar alguna tupla de una relación, ésta se transforma en otra distinta, pero con las mismas características.

Ejemplos:

- Relación de proveedores el día 1 de enero:

Núm. proveedor	*Nombre*
100	Pepe
200	Paco

- Relación de proveedores el día 1 de febrero:

Núm. proveedor	*Nombre*
100	Pepe
200	Paco
300	Pablo

Aquellas propiedades de una relación que no varían a lo largo del tiempo se conocen con el nombre de INTENSION. Dicho de otro modo, la intensión viene definida por todas las propiedades comunes a toda la familia de relaciones obtenibles al actualizar una dada.

Estas características invariantes al actualizar una relación son las siguientes:

- El nombre de la relación.
- Los dominios.
- El producto cartesiano de dominios sobre el que se construye la relación.
- Los nombres de los atributos.

La intensión también la conoceremos como ESQUEMA de la relación. Llamaremos extensión de una relación a las tuplas que contiene en un momento dado, que pueden variar al actualizarla.

El esquema se representa poniendo el nombre de la relación, seguido por los nombres de los atributos entre paréntesis y separados por comas. Así por ejemplo, R(A, B, C) es el esquema de una relación R con tres atributos, A, B y C.

BASE DE DATOS

Llamaremos base de datos a un conjunto finito de esquemas de relaciones.

SIGNIFICADO DE UNA RELACION

En apartados precedentes llamábamos significado de un registro a la información que éste aporta a la persona que lo recibe, con lo que el significado es algo subjetivo, dependiente de la persona receptora. Vamos a restringir este concepto para hacerlo algo menos subjetivo.

Cada tupla de una relación indica que sus componentes están ligados entre sí por unas determinadas propiedades, que constituyen el significado de la relación.

Dicho de otra forma, las tuplas que están contenidas en una relación no se han extraído al azar del producto cartesiano de sus dominios, $D_1 \times D_2 \times \cdots \times D_k$, sino de acuerdo con su significado.

Este significado es asignado por el analista cuando define el esquema conceptual.

Ejemplo:

Sea el esquema EM(DNI, SS, SU, FI). Su significado, asignado por el analista, podría venir definido por las propiedades siguientes:

> «Cada fila de la tabla EM se refiere a un empleado de la empresa. La columna DNI contiene el DNI del empleado. La columna SS su número de Seguridad Social. La SU su sueldo. La FI su fecha de ingreso en la empresa.»

Estas condiciones hacen que no sea válido incluir en EM tuplas con valores cualesquiera, sino sólo si éstos corresponden con los de un empleado.

La información que una tupla determinada aporta a un usuario viene definida en parte por el significado de la relación y en parte por los valores de sus atributos.

Asignar significado a un esquema de relación no es imprescindible desde un punto de vista mecanicista del modelo relacional, pues aun cuando las tuplas que se incluyeran en una relación se hubieran elegido al azar, se seguiría teniendo una relación formalmente válida.

Existen intentos de extensión del modelo relacional básico para que el sistema capture y gestione algo más del significado de las relaciones.

SIGNIFICADO DE LOS DOMINIOS

En el modelo relacional básico, de acuerdo con la definición dada anteriormente, un dominio es un conjunto, finito o infinito, de palabras formadas con un alfabeto finito entre las que existe un criterio de orden. Este criterio de orden puede ser común para varios dominios, por lo que éstos pueden participar unos con otros en ciertas operaciones que impliquen comparación y orden. Por ejemplo, tiene sentido hablar de unión, intersección, etc., entre ellos.

Se podría concebir un modelo relacional más avanzado en el que los dominios tuvieran un significado. Por ejemplo:

D_1: Conjunto de fechas de nacimiento.
D_2: Conjunto de los sueldos en pesetas de los empleados.
D_3: Conjunto de los nombres de los empleados.

Entonces se podrían restringir las operaciones de comparación y orden a elementos y subconjuntos de un mismo dominio. Así se aseguraría evitar hacer operaciones sin sentido (por ejemplo comparar fechas con sueldos). Nosotros no lo consideraremos así, mientras no se diga explícitamente lo contrario.

2 Algebra relacional

DEFINICION

El álgebra relacional es un sistema cerrado de operaciones definidas sobre relaciones. Es decir, tanto los operandos como los resultados son relaciones. Esto permite construir fórmulas o expresiones combinando unas operaciones con otras, de manera que los resultados de unas sean operando de otras.

Con más precisión:

Dado un conjunto de dominios, $D = (D_1, D_2, \ldots, D_n)$, sea R el conjunto de todas las relaciones posibles que se puedan construir sobre D. El álgebra relacional es un sistema formado por R y unos operadores cuyos operandos y resultados también pertenecen a R.

Vamos a describir a continuación estos operadores.

OPERADORES

Unión

$R \cup S$. (Como unión de conjuntos.)

El resultado es una relación que incluye a todas las tuplas de R y todas las de S, y ninguna más. Si hubiera alguna repetida en R y S, sólo figurará una vez en el resultado.

R y S deben ser relaciones del mismo grado

Diferencia

$R - S$. (Como diferencia de conjuntos.)

El resultado es una relación que incluye a todas las tuplas de R que no están en S.

R y S deben ser relaciones del mismo grado.

Intersección

R ∩ S. (Como intersección de conjuntos.)

El resultado es una relación que incluye a todas las tuplas de R que también están en S.

R y S deben ser relaciones del mismo grado.

Producto

R × S.

El resultado incluye a todas las tuplas posibles que se obtienen concatenando una de R con otra de S.

También lo llamaremos producto cartesiano de las relaciones R y S.

R y S pueden ser relaciones cualesquiera.

Proyección

P(A_1, A_2, ..., A_n) (R): Proyección de la relación R sobre los atributos $A_1, A_2, \ldots, A_n$.

El resultado se forma extrayendo de la relación R los atributos o columnas $A_1, A_2, \ldots, A_n$, y eliminando luego las tuplas que resulten repetidas, si las hay.

A veces, para abreviar, representaremos a una proyección, por ejemplo la **P**(X, Y, Z) (R), como **R**[X, Y, Z].

Selección

S(F) (R): Selección de la relación R de acuerdo con la fórmula F.

El resultado se forma extrayendo de la relación R todas las tuplas que cumplan las condiciones expresadas en la fórmula F.

Esta fórmula se construye con operandos que pueden ser constantes, o atributos de R, ligados entre sí por los operadores definidos sobre los respectivos dominios. Vamos a suponer que son operadores aritméticos (+, −, *, /, **), de comparación (>, <, =, >=, <=, ¬=), y lógicos (∨, ∧, ¬). (El significado de estos operadores lógicos se describe en uno de los apéndices.)

Ejemplo:

Si R tiene los atributos A, B, C y D, fórmulas válidas serían:

$A = 100$

$(A = 100) \wedge (B > 200)$

$(A = 100) \wedge (B > 200) \wedge (C + 8 < D ** 2)$

Con cierta frecuencia tendremos que referirnos a los valores de un atributo que corresponden a un valor fijo de otro. Para ello emplearemos la notación siguiente:

R(X = x, Y),

que representa parejas de valores formadas con todos los valores del atributo Y asociados a un determinado valor, x, de un atributo X, en una extensión de la relación R.

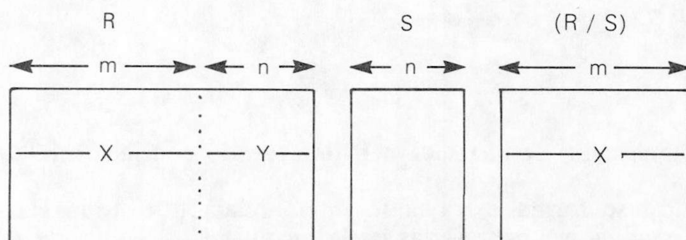
Esto lo podemos expresar más precisamente mediante las operaciones de proyección de la forma siguiente:

Si X e Y son atributos de R, (**R**(X = a, Y)) = (**P**(X, Y)(**S**(X = a)(R))). Normalmente no necesitaremos indicar que X es el atributo cuyo valor fijamos, sino que se podrá deducir del contexto. En este caso, la notación anterior se simplificará, quedando así: **R**(a, Y).

Cociente

Sean R y S relaciones de grado (m + n) y (n), respectivamente. Esto lo representaremos por $R_{(m+n}$, $S_{(n}$. Supongamos que S no es vacía.

Entonces, se define el cociente entre R y S, que lo representaremos por (R / S), como el conjunto de todas las tuplas de grado m tales que, al concatenarlas con todas las tuplas de S, producen tuplas contenidas en R.



Es decir, interpretando el dibujo anterior, se incluyen en (R / S) las partes izquierdas X de todas las tuplas de R tales que al hallar ((R / S) × S), se obtengan tuplas ⟨X, Y⟩ que están contenidas en R. O sea, ⟨X⟩ se incluye en (R / S) si y sólo si ⟨X⟩ × S ⊂ R (el signo ⊂ significa «contenido en»).

Se llama cociente por analogía:

```
((R / S)   ×    S)   ⊆   R
  :             :        :...... Dividendo
  :             :.. Divisor
  :.. Cociente
```

Yunción

Se representa R **Y** S(i **C** j).

Se lee: Yunción de R y S, mediante el operador de comparación **C** sobre las columnas i de R y j de S.

C es un operador de comparación (>, <, =, > =, < =, ¬ =).

i y j son atributos de R y S, respectivamente.

El resultado es el conjunto de todas las tuplas formadas concatenando una de R con otra de S de manera que se cumpla la condición (i **C** j).

Equiyunción: Cuando la comparación es por igualdad (o sea, el operador **C** es el signo =).

Yunción natural: La representaremos por (R **Y** S), o también por (R * S). (El signo * lo usamos también para representar la operación aritmética de multiplicar. El significado de * en cada caso vendrá definido por los operandos a los que se aplica. Si son relaciones, se tratará de una yunción natural, si son variables aritméticas, de una multiplicación. En todo caso, el contexto indicará claramente a cuál de ellos se refiere.)

Sean dos relaciones R y S que tienen algunos atributos iguales (es decir, columnas homónimas). La yunción natural es el resultado de:

1) Concatenar todas las tuplas de R y S. (es decir, hallar el producto R × S).

2) Seleccionar de entre estas tuplas concatenadas las que tengan iguales valores en todas sus columnas homónimas.

3) Suprimir una columna de cada dos homónimas en el resultado.

Si R y S no tienen atributos comunes, la yunción natural será igual al producto: R * S = R × S.

La yunción natural es asociativa y conmutativa. Esto nos permite definir la yunción natural para n operandos. Así tendremos:

R * S * T = (R * S) * T = R * (S * T) = (R * T) * S

2

Al ayuntar dos relaciones, no todas las tuplas presentes en los operandos aparecen en el resultado. Algunas pueden perderse. Dada

$$R = R_1 * R_2 * R_3 * \cdots * R_n,$$

diremos que todas las tuplas presentes en R_1, R_2, R_3,... R_n, también aparecen en R si las R_1, R_2,... R_n, puedan obtenerse proyectando R.

Ejemplos:

(Suponemos que todos los atributos están definidos sobre el mismo dominio.)

Unión, intersección, producto cartesiano, diferencia, proyección y selección:

R1	A	B	C
	a	b	c
	d	a	f
	c	b	d

S1	D	E	F
	b	g	a
	d	a	f

$R1 \cup S1$			
	a	b	c
	d	a	f
	c	b	d
	b	g	a

$R1 \cap S1$			
	d	a	f

$R1 \times S1$	A	B	C	D	E	F
	a	b	c	b	g	a
	a	b	c	d	a	f
	d	a	f	b	g	a
	d	a	f	d	a	f
	c	b	d	b	g	a
	c	b	d	d	a	f

$R1 - S1$			
	a	b	c
	c	b	d

P(A, C)(R1)	A	C
	a	c
	d	f
	c	d

S(B = b)(R1)	A	B	C
	a	b	c
	c	b	d

Cociente:

R2	A	B	C	D
	a	b	c	d
	a	b	e	f
	b	c	e	f
	e	d	c	d
	e	d	e	f
	a	b	d	e

S2	C	D
	c	d
	e	f

R2 / S2	A	B
	a	b
	e	d

Yunción:

R3	A	B	C
	1	2	3
	4	5	6
	7	8	9

S3	D	E
	3	1
	6	2

R3 Y S3 (B < D)	A	B	C	D	E
	1	2	3	3	1
	1	2	3	6	2
	4	5	6	6	2

Yunción natural:

R4	A	B	C
	a	b	c
	d	b	c
	b	b	f
	c	a	d

S4	B	C	D
	b	c	d
	b	c	e
	a	d	b

T4	A	C
	a	c
	c	d

R4 * S4	A	B	C	D
	a	b	c	d
	a	b	c	e
	d	b	c	d
	d	b	c	e
	c	a	d	b

R4 * S4 * T4	A	B	C	D
	a	b	c	d
	a	b	c	e
	c	a	d	b

(Puede verse en el ejemplo anterior que la tupla ⟨b b f⟩ de R4 no aparece en R4 * S4.)

2

POSIBILIDAD DE OPERACIONES DE ACTUALIZACION

Pueden usarse para ello los operadores de unión y diferencia.

Ejemplos:

1) Insertar la tupla $\langle 100, a, b \rangle$ en R:
   $R \cup (\langle 100, a, b \rangle)$
2) Borrar la tupla $\langle 100, a, b \rangle$ de R:
   $R - (\langle 100, a, b \rangle)$.

OPERACION DE COMPLEMENTO

No hemos definido en el álgebra relacional una operación de complemento, porque el conjunto de tuplas complementario de una relación R, no será, en general, otra relación de **R**.

Se podría haber intentado definirlo como una relación $(-R)$, tal que:

$$(-R) = (D_1 \times D_2 \times D_3 \times \cdots \times D_k) - (R)$$

(donde $D_1, D_2, \ldots, D_k$, son los dominios usados en R). Si alguno de los dominios es infinito, también lo será $(-R)$, por lo que ésta no será una relación válida.

No usaremos operaciones de complemento.

POTENCIA EXPRESIVA DEL ALGEBRA

Hemos incluido en el álgebra los operadores unión, intersección, diferencia, cociente, producto cartesiano, proyección, selección y yunción. Se podrían inventar más operadores, pero es innecesario si nos limitamos a tener en el álgebra tanto poder expresivo como en el cálculo, que se verá en el próximo capítulo.

El álgebra relacional es un lenguaje relacionalmente completo. Esto significa que tiene la misma potencia de expresión que el cálculo relacional. Es decir, cualquier fórmula de álgebra relacional es expresable en una sentencia de cálculo relacional, y viceversa.

Por tanto, no es necesario inventar más operadores, a no ser que queramos dotar al álgebra de más potencia expresiva que el cálculo. Por ejemplo, el lenguaje relacional QBE (Query By Example) permite algunas operaciones no expresables en cálculo o álgebra.

OPERACIONES PRIMITIVAS

Hemos incluido en álgebra ocho operadores: unión, intersección, diferencia, cociente, producto cartesiano, proyección, selección y Yunción ($\cup$, $\cap$, $-$, $/$, $\times$, **P**, **S**, **Y**).

No son todos ellos necesarios. Algunos pueden expresarse en función de otros.

Un conjunto de operadores primitivo, es decir, que ninguno de ellos puede expresarse en función de los otros, es el siguiente:

$\cup$, $-$, $\times$, **P**, **S**.

Veamos cómo expresar los otros tres en función de estos cinco.

Intersección:

$(R \cap S) = (R - (R - S))$.

Yunción:

$(R\ \mathbf{Y}\ S(i\ \mathbf{C}\ j)) = \mathbf{S}(i\ \mathbf{C}\ (r+j))(R \times S)$, (donde r es el grado de R).

División (R de grado $m+n$, S de grado n):

Sea $T = \mathbf{P}(1, 2, \ldots, m)(R)$. Entonces:
$(R / S) = T - \mathbf{P}(1, 2, \ldots, m)((T \times S) - R)$.

PROPIEDADES DE LOS OPERADORES

Ya hemos visto cómo algunos operadores pueden expresarse en función de otros. Un estudio más a fondo de las propiedades de los operadores es prerrequisito para poder optimizar la evaluación de expresiones. Esto nos plantea dos problemas:

1) transformar una expresión en otra equivalente;

2) determinar si la equivalente puede evaluarse en un tiempo más corto.

Para resolver el primero hay que conocer las propiedades de los operadores. Citaremos, a modo de ejemplo:

1) $\cup$ es conmutativo y asociativo:
$(A \cup B) \cup C = A \cup (B \cup C)$.

2) Análogamente $\cap$.

3) Propiedad distributiva de $\cup$ y $\cap$:
$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$.

4) Propiedad conmutativa de la selección y el producto cartesiano:
$\mathbf{S}(F1 \wedge F2)(A \times B) = (\mathbf{S}(F1)(A)) \times (\mathbf{S}(F2)(B))$,
donde la fórmula F1 sólo se refiere a los atributos de A y la F2 a los de B.

Dos expresiones son equivalentes cuando dan el mismo resultado para todos los valores posibles de las variables.

Entre varias expresiones equivalentes elegiremos la que pueda evaluarse en menos tiempo, es decir, empleando menos operaciones elementales.

Ejemplo:

Sea
R(A, B)
S(B, C)
T1 = P(C)(S(A = a)(R * S)).

Esta expresión es equivalente a:

T2 = P(C)((P(B)(S(A = a)(R))) * S)

Sin embargo, la expresión T2 es preferible, pues, en principio, se evaluará en menos tiempo que T1.

Importancia de la optimización

Gracias al proceso de formalización se consigue poder expresar en fórmulas los procesos que sobre ficheros tradicionales secuenciales se reflejaban en organigramas con varios pasos (clasificación, enfrentamiento de maestro con movimiento, selección). Y además, el sistema puede optimizar estos procesos (antes esta labor tenía que hacerse basándose en la intuición y la experiencia).

Importancia del cálculo (o el álgebra)

Son los lenguajes patrón, que permiten comparar la potencia expresiva de otros.

Además, si las sentencias de un lenguaje son expresables en cálculo (o álgebra), inmediatamente le son aplicables los algoritmos de optimización desarrollados para aquél.

VISTAS

Otra consecuencia importante de la estructura de sistema cerrado que tiene el álgebra es el concepto y utilización de vistas.

El que los operandos de una expresión puedan a su vez ser expresiones, permite tener expresiones prefabricadas que se usan para construir otras más complejas.

Se llama vista a una expresión o fórmula algebraica a la que se le asigna un nombre. Este nombre puede utilizarse como operando en la construcción de nuevas fórmulas, a las que también se les puede asignar un nombre a su vez.

El uso de vistas puede aplicarse para:

- Proteger la confidencialidad de acceso a los datos.
- Simplificar la formulación de expresiones complejas.
- Evitar que cambios en el diseño afecten a los programas ya existentes.

Ejemplo 1:

Consideremos las relaciones EMP(NUEM, NOME, SUELDO, NUDE) y DEP(NUDE, NOMDE).

EMP: Contiene una tupla por cada empleado.
DEP: Contiene una tupla por cada departamento.
NUEM: Número identificador de empleado (clave).
NOME: Nombre de empleado.
NUDE: Número de departamento (clave de DEP).
NOMDE: Nombre de departamento.

Supongamos que un usuario desea acceder a los datos de los empleados, pero no queremos permitirle que acceda a los datos de sueldos. Para ello podemos definir la vista siguiente:

EMPUBLI = **P** (NUEM, NOME, NUDE) (EMP)

Si ahora autorizamos al usuario a emplear en la construcción de fórmulas la vista EMPUBLI, pero no la tabla EMP, estaremos impidiéndole el acceso a los datos confidenciales de sueldos.

Ejemplo 2:

Queremos hallar, con las relaciones del ejemplo anterior, los nombres de los empleados que trabajan en el departamento cuyo nombre es DA.

Para ello podemos definir la vista EMPDEP:

EMPDEP = EMP * DEP

Los datos de los empleados del departamento DA vienen dados por la vista:

V1 = **S** (NOMDE = DA) (EMPDEP)

Sus nombres vienen dados por:

P (NOME) (V1)

En definitiva, hemos construido gradualmente la fórmula:

P (NOME) (S(NOMDE = DA) (EMP * DEP))

Ejemplo 3:

Sea la relación PED (AR, PR, CA, NP).

PED: Contiene una tupla por cada pedido.
AR: Número de Artículo.
PR: Número de Proveedor.
CA: Cantidad pedida.
NP: Nombre del Proveedor.
Clave: (AR, PR).

Decidimos cambiar este diseño sustituyendo la relación PED por las siguientes:

PEDI (AR, PR, CA), (Clave: (AR, PR))
PROV (PR, NP), (Clave: PR)

Entonces todas las fórmulas en las que se haya empleado PED dejan de ser válidas. Sin embargo, si definimos la siguiente vista (con el mismo nombre), siguen siendo válidas:

PED = PEDI * PROV

Es decir, los programas que usaban la tabla PED original en modo lectura (o sea para consultas), siguen siendo válidos si usan la nueva vista. Los programas que actualizaban la tabla PED original pueden no ser válidos al intentar trabajar con la vista PED. Bajo qué condiciones serán válidos o no constituye un problema que se sale de nuestros objetivos, por lo que no vamos a entrar en él.

VALORES NULOS

El permitir que algunos atributos puedan tomar valores nulos tiene efectos importantes en las definiciones anteriores, que deben ser revisadas y adaptadas. Vamos a tratar de exponer brevemente algunos de ellos.

Se suelen emplear los valores nulos para representar información incompleta. Representaremos los valores nulos con el signo ?.

Ejemplo:

Si en la relación EMPLEADOS (NOMBRE, EDAD), quisiéramos dar de alta al empleado PEPE sin saber su edad, insertaríamos la tupla ⟨PEPE, ?⟩. De modo que el valor nulo, ?, puede interpretarse como «desconocido».

Una primera consideración a tener en cuenta es que puede haber distintos tipos de valores nulos en este sentido de información incompleta. En efecto, un valor puede ser menos «desconocido» que otro, en el sentido de que hay veces que tenemos sobre él alguna información, aunque incompleta; por ejemplo, si sabemos que el valor debe estar en un determinado intervalo, aunque no sepamos el valor exacto. Sin embargo, nosotros vamos a representar a todos por un mismo signo, ?, y no distinguiremos entre estos distintos niveles de conocimiento.

Ejemplos:

En el ejemplo anterior, aunque sepamos que el empleado PEPE tiene menos de 25 años, seguiremos representándolo por la misma tupla, ⟨PEPE, ?⟩, que si no supiéramos nada sobre su edad.

Veamos otro ejemplo. Sea la relación PED(NA, NP, DA), donde hay una fila por pedido, NA representa el número del artículo pedido, NP el del proveedor y DA la descripción del artículo. Evidentemente, todas las filas que se refieran a pedidos del mismo artículo deberán tener la misma descripción DA. Supongamos que pedimos los artículos A1 y A2, cuyas descripciones no conocemos, al proveedor P1. Esto se reflejará insertando las tuplas ⟨A1, P1, ?⟩, y ⟨A2, P1, ?⟩. Estos dos valores nulos representan valores desconocidos, pero que tienen que ser diferentes. Sin embargo, empleamos un único signo, ?, para ambos.

El concepto de dominio debe ser revisado. Para los atributos que puedan tomar valor nulo, incluiremos en su dominio el signo ?. Puesto que el dominio es un conjunto, consideraremos el signo ? como un elemento más del conjunto, distinto a todos los demás.

Una relación es un conjunto de tuplas. Por tanto, no puede haber tuplas duplicadas. Dadas dos tuplas, consideraremos que una es duplicada de la otra cuando todos los atributos que son nulos en una lo son también en la otra, y

todos los que no son nulos tienen en la otra el mismo valor. Así, la tupla ⟨PEPE, ?⟩ y la ⟨PEPE, 25⟩ no son duplicadas y, por tanto, podrían coexistir en la relación EMPLEADOS.

El concepto de superclave debe ser retocado. Diremos que un atributo A es una superclave cuando sus valores no nulos son todos distintos. Si A es compuesto, intrepretaremos que no es nulo cuando todos sus componentes no lo sean.

Las operaciones con atributos deben ser revisadas para definir cómo operar con nulos. Así, se puede establecer la regla de que cualquier operación aritmética con un operando nulo dé resultado nulo. Esto es coherente con la interpretación de valor desconocido que hemos dado al nulo (el resultado de operar con un valor desconocido es desconocido). Las operaciones de comparación entre dos valores, que normalmente pueden dar resultado verdadero o falso, ahora podrán dar también desconocido o nulo, en caso de que alguno de los comparandos sea nulo. Por tanto, necesitaremos aplicar una lógica de tres valores (V, F, ?) en vez de la de dos valores, más sencilla e intuitiva. Esto se extiende a expresiones en que las comparaciones se combinan con operadores lógicos ($\wedge$), ($\vee$), ($\neg$). Las tablas de resultados para estos operadores serán:

$\wedge$	V	?	F
V	V	?	F
?	?	?	F
F	F	F	F

$\vee$	V	?	F
V	V	V	V
?	V	?	?
F	V	?	F

$\neg$	
V	F
?	?
F	V

Aunque la comparación de un valor cualquiera con otro nulo dé resultado nulo, para poder ordenar los elementos de un dominio es necesario establecer entre ellos un criterio bien definido de orden, incluyendo al nulo, ?.

Hay que incluir también un nuevo predicado o condición que nos permita preguntar si el valor de un atributo es nulo. Lógicamente, sólo podrá dar como resultado V o F (no podrá dar nulo, ?; es decir, no puede ser desconocido saber si toma el valor desconocido).

En cuanto a las operaciones del álgebra relacional, un ejemplo de las que habría que ampliar es la yunción. Si tomamos el criterio de considerar como iguales entre sí dos valores nulos, o un valor nulo y cualquier otro, obtenemos la yunción posibilista («may - be - join»).

Ejemplo:

R	A	B
	a	1
	b	2
	c	?
	?	3

S	B	C
	1	A
	?	B
	2	?
	4	C

R YPS	A	R.B	S.B	C
	a	1	1	A
	a	1	?	B
	b	2	2	?
	b	2	?	B
	c	?	1	A
	c	?	?	B
	c	?	2	?
	c	?	4	C
	?	3	?	B

2

Es necesario definir una nueva operación de yunción natural, llamada yunción natural externa. Es como la yunción natural ya vista, pero añadiendo al resultado las tuplas de los operandos que, o bien tengan ? en el atributo común, o bien éste no exista en la otra relación. Estas tuplas añadidas se completan con ?.

Yunción natural externa (YE):

R	A	B
	a	1
	b	2
	?	3
	c	4
	d	5

S	A	C
	a	6
	e	7
	?	8
	c	9
	f	A

R YE S	A	B	C
	a	1	6
	b	2	?
	?	3	?
	c	4	9
	d	5	?
	e	?	7
	?	?	8
	f	?	A

(Obviamente: $(R * S) \subseteq (R \text{ YE } S)$).

Equiyunción externa (con los mismos datos R y S anteriores):

R YE S(R.A = S.A)	R.A	B	S.A	C
	a	1	a	6
	b	2	?	?
	?	3	?	?
	c	4	c	9
	d	5	?	?
	?	?	e	7
	?	?	?	8
	?	?	f	A

Unión externa (suponemos que los atributos homónimos están definidos sobre el mismo dominio, y los heterónimos sobre dominios diferentes):

R	A	B	C
	a1	b1	c1
	a1	b1	c5
	a2	b2	c1

S	B	D
	b1	d1
	b3	d2

R UE S	A	B	C	D
	a1	b1	c1	?
	a1	b1	c5	?
	a2	b2	c1	?
	?	b1	?	d1
	?	b3	?	d2

En resumen, la inclusión de valores nulos exige una revisión y adaptación de las definiciones básicas y de las operaciones algébricas, según se ha expuesto.

Nosotros admitiremos la posibilidad de valores nulos, pero sin revisar todos los aspectos que aquí se han indicado. Esto lo precisamos en el apartado siguiente.

MODELO RELACIONAL DE DATOS

Un modelo de datos es un sistema formal y abstracto que permite describir los datos de acuerdo con unas reglas y convenios predefinidos. Es formal en el sentido de que los objetos del sistema se manipulan siguiendo unas reglas perfectamente definidas, y utilizando exclusivamente los operadores definidos en el sistema, independientemente de lo que estos objetos y operadores puedan significar.

Por el contrario, el modelo conceptual de datos, que como ya se ha dicho, es el conjunto de conceptos e interrelaciones que en la mente del analista forman una imagen del mundo real, no es un sistema formal.

El modelo de datos es el lenguaje en el que el analista describe el modelo conceptual que su mente ha concebido, llamándose esta descripción, como ya se ha dicho, esquema conceptual.

Un modelo de datos es tanto mejor cuanto más capacidad expresiva tenga para reproducir fielmente en el esquema conceptual el comportamiento del modelo conceptual, que a su vez debe ser imagen fiel del mundo real si está bien concebido.

Un modelo de datos tiene tres componentes:

1) *Estructura de datos:* Es una colección de objetos abstractos formados por datos.

2) *Operadores entre las estructuras:* Conjunto de operadores, con reglas bien definidas, que permiten manipular las estructuras de datos.

3) *Definiciones de integridad:* Colección de conceptos y reglas que permite expresar qué valores de datos pueden aparecer válidamente en nuestro esquema.

Existen varios modelos de datos generalmente aceptados. Entre ellos los más utilizados son el jerárquico, el modelo en red y el relacional. Este último es el que aquí nos interesa.

La definición del modelo relacional de datos generalmente aceptada en la literatura sobre el tema incluye la posibilidad de nulos, lo que lleva a redefinir los conceptos de clave, dominio, etc., en la forma en que se ha establecido en el apartado anterior. Sin embargo, no suelen incluirse en él las operaciones algébricas extendidas (por ejemplo, yunción externa y unión externa). Este será el modelo de datos al que nos referiremos de ahora en adelante. En resumen, tiene los componentes siguientes:

1) *Estructura de datos:* Dominios, relaciones, atributos, tuplas.

2) *Operadores:* Los primitivos del álgebra relacional, es decir, unión, diferencia, producto cartesiano, proyección y selección.

3) *Definiciones de integridad:* Los conceptos de claves y la posibilidad de valores nulos.

También se incluyen aquí dos reglas de integridad, llamadas:

a) Integridad de claves primarias.

b) Integridad referencial.

Ya se han comentado previamente todos estos componentes del modelo relacional de datos, excepto las reglas de integridad. Vamos a describirlas a continuación.

Integridad de claves primarias

Al definir el concepto de claves se dijo que éstas pueden usarse como identificadores de las tuplas de una relación, puesto que a cada valor de una clave corresponde una sola tupla y viceversa.

En el modelo relacional, la única manera de encontrar una tupla determinada en una relación, es conociendo el valor de una clave.

Una relación puede tener varias claves, pero suele aceptarse la conveniencia de emplear siempre la misma como identificador. A esta clave se la suele llamar clave primaria. Las restantes se llaman claves alternativas.

Puesto que la clave primaria es el identificador designado para una relación, no debería de tomar valores nulos para evitar ambigüedades. Esta condición es la que hemos llamado regla de integridad de claves primarias (en la literatura se la suele llamar integridad de entidad).

Con más precisión, la enunciamos así: ningún atributo de una clave primaria puede tomar valores nulos.

Ejemplo:

Veamos un caso de la ambigüedad que podría surgir si no se cumpliera esta regla. Consideremos la relación EMP (NE, ND, SUELDO), donde cada tupla representa a un empleado de una empresa, NE es el número de un empleado, ND el departamento en que trabaja y SUELDO su sueldo. Tiene una sola clave: NE, que, por tanto, será la clave primaria.

Si NE pudiera tomar valores nulos, podrían existir en EMP las tuplas:

EMP	(NE	ND	SUELDO)
	E1	D1	1000
	?	D1	1000

La segunda tupla (en el orden en que se han escrito), se refiere a un empleado desconocido. ¿Tenemos, por tanto, un solo empleado o dos? Imposible saberlo con la información aportada por la relación solamente.

Podría pensarse que en el caso de tener varias claves, podríamos emplear como identificador unas veces a una y otras a otra, con lo que no tendríamos una única clave primaria, y ambas podrían tomar valores nulos. Esto no es así, y puede crear problemas de ambigüedad.

Ejemplo:

Sea la relación anterior, EMP, con un atributo más, NSS, que representa el número de afiliado a la Seguridad Social. La relación es por tanto: EMP (NE, NSS, ND, SUELDO). Tiene dos claves: NE y NSS. Supongamos que ambas pudieran tomar valores nulos (no simultáneamente). Podrían existir en EMP las tuplas:

EMP	(NE	NSS	ND	SUELDO)
	E1	?	D1	1000
	?	SS1	D1	1000

De nuevo es imposible contestar a la pregunta de si ambas tuplas representan al mismo empleado o no.

En conclusión, en toda relación debe de designarse a una clave como primaria, y sus atributos no deben de tomar valores nulos.

Si una relación tiene varias claves, cualquiera de ellas puede ser designada a priori como primaria. Para elegir la más conveniente, el diseñador de la Base de Datos deberá tener en cuenta el significado de la relación y sus atributos, y sopesar diversos factores. Pueden considerarse los siguientes:

- Estabilidad.
  Considerar aquí si algunas claves son menos propensas a sufrir modificaciones en sus valores.
- Facilidad de uso.
  Será, por ejemplo, más fácil de usar una clave numérica corta que otra alfanumérica con muchos caracteres.
- Fiabilidad.
  Ver si alguna clave contiene dígitos de validación u otros mecanismos de autodetección o corrección de errores.
- Universalidad.
  Puede haber claves cuyo uso y conocimiento esté muy extendido (por ejemplo el número del Documento Nacional de Identidad).

Integridad de referencia

Es posible que unas relaciones hagan referencia a otras por medio de las claves primarias de éstas. Vamos a analizar esto más detalladamente.

Ejemplo:

Sean la relación EMP anterior, EMP (NE, NSS, ND, SUELDO), con clave primaria NE, y la DEP (ND, NOMBRE), donde cada tupla representa un departa-

mento de nombre NOMBRE, con clave primaria ND. NSS y NOMBRE son claves alternativas. Vemos que las tuplas de EMP hacen referencia a las de DEP por medio de la clave primaria de DEP, ND. Estas referencias entre relaciones suelen implicar condiciones de existencia. En nuestro ejemplo, no puede haber un empleado que trabaje en un departamento del que no tenemos información, lógicamente. Dicho de otra forma, todo valor de ND en EMP debe existir en DEP. Cuando un atributo de una relación es clave de otra, y toma valores contenidos en ésta, se le suele llamar clave ajena. En nuestro ejemplo, ND es una clave ajena en EMP, y clave primaria en DEP. La regla de integridad referencial es que todo valor de la clave ajena debe de existir en una clave primaria. Enunciemos estas definiciones con más precisión.

Se dice que un atributo, A, de una relación, R, es una clave ajena si se requiere que todos los valores de A no nulos existan en alguna clave primaria de alguna relación, no necesariamente distinta de R. Si A es un conjunto de atributos, $A_1, A_2, \ldots, A_n$, la definición anterior sigue siendo válida si se interpreta que A toma valor nulo cuando uno cualquiera de sus componentes sea nulo.

Cuando en el esquema conceptual se designa a un atributo como clave ajena, conviene también especificar las acciones a tomar en caso de intentar actualizarlo con valores inválidos. Estas acciones dependerán del significado de los datos.

Ejemplo:

Así, en las actualizaciones de las relaciones anteriores, EMP (NE, NSS, ND, SUELDO) y DEP (ND, NOMBRE) pueden presentarse los siguientes casos. En cada uno de ellos se indican las posibles acciones, entre las que se debería elegir una, en función de cómo se asignen los empleados a los Departamentos.

1) *Inserción de una nueva tupla en DEP.* No hay que comprobar nada.

2) *Borrado de una tupla de DEP.* Hay que comprobar que el departamento que borramos no tiene empleados. Si así no fuere, la acción a tomar para mantener los datos en un estado válido de acuerdo con la integridad de referencia, podría ser una de las siguientes:
   a) Rechazar la petición de borrar la tupla de DEP.
   b) Aceptarla y propagarla a EMP. Es decir, borrar en EMP todas las tuplas que hagan referencia al departamento borrado. Este borrado podría propagarse a su vez a otras relaciones que hagan referencia a EMP (propagación en cascada).
   c) Aceptarla y anular la referencia. Es decir, actualizar todas las tuplas de EMP que hagan referencia al departamento borrado poniendo nulo en el atributo EMP.ND.

3) *Actualización de una tupla de DEP modificando al atributo ND.* Hay que comprobar que el departamento que modificamos no tiene empleados. Si así no fuere, la acción a tomar para mantener los datos en un estado válido de acuerdo con la integridad de referencia, podría ser una de las siguientes:
   a) Rechazar la petición de actualización.
   b) Aceptarla y propagarla a EMP. Es decir, actualizar en EMP todas las tuplas que hagan referencia al departamento modificado poniendo el nuevo valor en EMP.ND.
   c) Aceptarla y anular la referencia. Es decir, actualizar todas las tuplas de EMP que hagan referencia al departamento borrado poniendo nulo en el atributo EMP.ND.

4) *Inserción de una nueva tupla en EMP.* Hay que comprobar que el departamento de este nuevo empleado existe en DEP si no es nulo. Si así no fuere, se rechazará la petición de insertar la nueva tupla en EMP.

5) *Borrado de una tupla de EMP.* No hay que comprobar nada.

6) *Actualización de una tupla de EMP modificando al atributo ND.* Hay que comprobar que si el nuevo valor de EMP.ND no es nulo, existe en DEP. Si así no fuere, se rechazará la petición de actualización.

EJERCICIOS PROPUESTOS

1) *Sean las relaciones R y S siguientes:*

R	A	B
	a	b
	c	b
	d	e

S	B	C
	b	c
	b	d
	e	a

Hallar los resultados de las siguientes expresiones:

1.1) R ∪ S.

1.2) R − S.

1.3) R ∗ S.

1.4) **P**(A)(R).

1.5) **S**(A = C)(R × S).

2) *Sean los esquemas de relación siguientes:*

a) HOMBRES (NOMH, EDAD)
Clave: (NOMH)
Significado: Cada fila representa un hombre, cuyo nombre es NOMH y su edad en años viene en EDAD.

b) MUJERES (NOMM, EDAD)
Clave: (NOMM)
Significado: Cada fila representa una mujer, cuyo nombre es NOMM y su edad en años viene en EDAD.

c) HSIM (NOMH, NOMM)
Clave: (NOMH, NOMM)
Significado: El hombre NOMH cae simpático a la mujer NOMM.

d) MSIM (NOMH, NOMM)
Clave: (NOMH, NOMM)
Significado: La mujer NOMM cae simpática al hombre NOMH.

e) MATRIM (NOMH, NOMM)
Clave: (NOMH, NOMM)
Significado: La pareja NOMH y NOMM están casados.

2

Escribir las sentencias necesarias para responder a las preguntas siguientes.

2.1) Hallar las parejas de hombres y mujeres que se caen mutuamente simpáticos.

2.2) Hallar las parejas de hombres y mujeres que se caen mutuamente simpáticos, con edades entre 20 y 30 años y que no estén casados entre sí.

2.3) Hallar las parejas casadas cuyos componentes se caen mutuamente simpáticos.

2.4) Hallar las mujeres casadas a quienes no cae simpático su marido.

2.5) Hallar los hombres misóginos a quienes no cae simpática ninguna mujer.

2.6) Hallar los hombres y mujeres asociales a quienes no cae nadie simpático.

2.7) Hallar las mujeres casadas que caen simpáticas a algún hombre.

2.8) Hallar los hombres a quienes sólo caen simpáticas mujeres casadas.

3) *Sean las relaciones siguientes:*

a) SOCIO (AFICIONADO, VIDEOCLUB)
Significado: AFICIONADO es SOCIO de VIDEOCLUB.

b) GUSTA (AFICIONADO, PELICULA)
Significado: PELICULA de cine GUSTA a AFICIONADO.

c) VIDEOTECA (VIDEOCLUB, PELICULA)
Significado: VIDEOCLUB dispone en su VIDEOTECA de PELICULA.

Escribir las sentencias necesarias para responder a las preguntas siguientes.

3.1) Videoclubes que disponen de alguna película que le guste al aficionado José Pérez.

3.2) Aficionados que son socios al menos de un videoclub que dispone de alguna película de su gusto.

3.3) Aficionados que son socios solamente de videoclubes que disponen de alguna película de su gusto.

3.4) Aficionados que no son socios de ningún videoclub donde tengan alguna película de su gusto.

4) *Sean las tablas siguientes:*

a) PRO (NP, NOMP, CIUDADP)
Clave: (NP)
Significado: Cada fila representa un proveedor, cuyo identificador es NP, su nombre NOMP, y habita en la ciudad CIUDADP.

b) ART (NA, DESA, COLOR, TALLA)
Clave: (NA)
Significado: Cada fila representa un artículo, cuyo identificador es NA y su descripción DESA.

c) FAB (NF, NOMF, CIUDADF)
Clave: (NF)
Significado: Cada fila representa una fábrica, cuyo identificador es NF, su nombre NOMF, y está situada en CIUDADF.

d) PED (NP, NA, NF, CANTIDAD)

Clave: (NP, NA, NF)

Significado: Cada fila representa un pedido del artículo NA, al proveedor NP, para la fábrica NF.

Escribir las sentencias necesarias para responder a las preguntas siguientes:

4.1) Hallar los identificadores de todas las fábricas.

4.2) Hallar los nombres de las fábricas situadas en Madrid.

4.3) Artículos tales que ningún otro tiene talla más pequeña.

4.4) Proveedores que suministran a la fábrica F1.

4.5) Proveedores que suministran a la fábrica F1 el artículo A1.

4.6) Nombres de las fábricas a las que suministra el proveedor P1.

4.7) Colores de los artículos suministrados por el proveedor P1.

4.8) Qué proveedores suministran a las fábricas F1 y F2.

4.9) Proveedores que suministran artículos azules a la fábrica F1.

4.10) Artículos suministrados a las fábricas de Madrid.

4.11) Proveedores que suministran algún artículo azul a las fábricas de Madrid o Lisboa.

4.12) Artículos suministrados por proveedores en cuya ciudad hay alguna fábrica.

4.13) Artículos suministrados a las fábricas de Madrid por proveedores de Madrid.

4.14) Fábricas abastecidas por al menos un proveedor de distinta ciudad.

4.15) Fábricas que no son abastecidas de artículos azules por proveedores de Madrid.

4.16) Proveedores que suministran al menos un artículo suministrado por al menos otro proveedor que suministra al menos un artículo azul.

4.17) Fábricas que usan al menos un artículo suministrado por el proveedor P1.

4.18) Parejas de ciudades tales que un proveedor de la primera abastece a una fábrica de la segunda.

4.19) Obtener las tripletas de valores de ⟨CIUDAD, NA, CIUDAD⟩ tales que un proveedor de la primera ciudad abastece el artículo NA a una fábrica de la segunda ciudad.

4.20) Repetir la pregunta anterior, pero sin obtener las tripletas en que ambas ciudades son la misma.

4.21) Proveedores que suministran un mismo artículo, al menos, a todas las fábricas.

4.22) Fábricas que tienen como único proveedor a P1.

4.23) Artículos que son suministrados a todas las fábricas de Madrid.

4.24) Fábricas que usan, al menos, todos los artículos suministrados por el proveedor P1.

4.25) Fábricas que usan sólo artículos que pueden ser suministrados por el Proveedor P1.

4.26) Fábricas abastecidas por el proveedor P1 con todos los artículos que éste suministra.

4.27) Fábricas que obtienen del proveedor P1, total o parcialmente, todos los artículos que usan.

4.28) Fábricas abastecidas por todos los proveedores que suministran algún artículo de color azul.

5) *Sean las relaciones siguientes:*

a) EMPM (#E, #D, NOME, CAT, SUELDO)

Hay una tupla por cada empleado destinado en Madrid.

#E: Número de empleado.

#D: Número de departamento en el que trabaja.

NOME: Nombre del empleado.

CAT: Categoría del empleado.

SUELDO: Sueldo del empleado.

Clave primaria: #E.

b) EMPL (#E, #D, NOME, CAT, SUELDO)

Hay una tupla por cada empleado destinado en Lisboa.

Iguales atributos que EMPM.

Todos los empleados de la empresa están, o bien en EMPM, o bien en EMPL. Un empleado no puede estar a la vez en EMPM y EMPL.

Clave primaria: #E.

c) DEP (#D, NOMD, JEFE, ORG, LOCAL)

Hay una tupla por cada departamento de la empresa, sea cual sea la ciudad donde se encuentre (Madrid o Lisboa).

#D: Número de departamento.

NOMD: Nombre del departamento.

JEFE: Número de empleado del director del departamento.

ORG: Número de departamento del que este departamento depende. Se supone que en el organigrama de la empresa, cada departamento depende de otro, y solamente de él, en una estructura jerárquica. Naturalmente habrá de existir un departamento origen, que no depende de ningúno.

LOCAL: Dirección (es decir, ciudad, calle, número, planta) donde se encuentra ubicada la oficina del departamento. Algunos de estos locales son propiedad de la empresa y otros están alquilados.

Claves: #D, NOMD, JEFE.

Clave primaria: #D.

d) ALQ (LOCAL, CANTIDAD)

Hay una tupla por cada local alquilado. Cada local debe contener al menos un departamento.

LOCAL: El mismo significado que en DEP.

CANTIDAD: Coste del alquiler del local.

Clave primaria: LOCAL.

Preguntas:

a) ¿Qué atributos no deben de tomar valores nulos de acuerdo con la regla de integridad de claves? (Integridad de Entidad).

b) ¿Qué atributos son claves ajenas? (Integridad de Referencia).

Cálculo relacional

3

INTRODUCCION INFORMAL AL CALCULO DE TUPLAS

El cálculo relacional es un lenguaje basado en cálculo de predicados de primer orden (recuérdese que una relación expresa una propiedad o predicado), que es un intento de reproducir las leyes del razonamiento humano y, por tanto, de captar una parcela reducida del lenguaje natural.

Es un lenguaje no procedimental («non procedural»). Es decir, se expresa en él lo que se quiere obtener, no cómo obtenerlo.

Notación

Conjunto de dominios:

$\mathbf{D} = (D_1, D_2, D_3, \ldots, D_n)$.

Conjunto de todas las tuplas posibles que se pueden formar sobre **D** : **TUP**.

Variables que representan tuplas de grado m, n, p, ... (puede omitirse el grado si está claramente implícito en el contexto):

$t_{(m}, u_{(n}, v_{(p}, \ldots$

Estas variables toman valores en **TUP** o en subconjuntos de **TUP**. El conjunto de todos los valores posibles de una variable, t, lo representaremos por DOM(t).

Mientras no se diga lo contrario, DOM(t) será el conjunto **TUP**.

Para designar a los componentes de una tupla t emplearemos la notación:

t(1), t(2), t(3), ...

O también, a veces, si no se presta a confusión:

$t_1, t_2, t_3, \ldots$

Relaciones definidas sobre **D**: letras mayúsculas superiores, como por ejemplo R, S, T, ...

Conjunto de todos los atributos de una relación dada: U.

Atributos: Las letras mayúsculas inferiores suelen representar atributos simples, por ejemplo A, B, C, ... Las últimas del alfabeto suelen representar atributos compuestos (conjuntos de atributos simples), por ejemplo X, Y, Z. Sin embargo, a veces, los atributos simples se representarán con un subíndice, como por ejemplo A_1, A_2, A_3, ..., utilizando en este caso cualquier letra sin subíndice para atributos compuestos.

Condiciones de comparación

Se construyen combinando variables y constantes con operadores de comparación y aritméticos.

Comparan componentes de tuplas o tuplas enteras. Toman el valor de verdadero, falso o nulo, según sea el resultado de la comparación.

Ejemplos:

- Componentes de una misma tupla: $(t_1 > t_2)$.
- Componentes de distintas tuplas: $(t_1 \neg = u_1)$.
- Constantes: $(t_3 = 8)$.
- Con operación aritmética: $(t_1 = t_2 + u_2)$.

Condiciones de pertenencia

Se representan en la forma R(t).

La condición R(t) toma el valor verdadero si la tupla t pertenece a la relación R, y falso en caso contrario.

Condiciones compuestas

Se construyen a partir de las anteriores combinándolas con operadores lógicos ($\wedge$), ($\vee$) y ($\neg$).

Ejemplos:

$(t_1 > t_2) \wedge (t_1 > 8)$.

$(R(t) \wedge (t_1 = 8))$

$(R(t) \wedge (\neg S(t)) \wedge (t_1 = 8))$

Estas expresiones pueden ser V, F ó ?, según sean los valores de la tupla que sustituye a t (es decir, según el valor que tome la variable t).

Cuantificador existencial

Lo representaremos como: $\exists$t (condición).

Esto se lee: «Existe alguna tupla t que cumpla la condición.»

La fórmula

$\exists$t (condición)

es verdadera si existe al menos un valor de la variable t que cumpla la condición, y es falsa en caso contrario.

Ejemplos:

1) La fórmula

$$\exists t(R(t))$$

es verdadera si R no está vacía.

2) La fórmula

$$\exists t(R(t) \wedge (t_1 = 8)).$$

es verdadera si alguna tupla de R empieza por 8.

A veces, para simplificar la notación, restringiremos alguna variable a un determinado conjunto de valores.

Ejemplo:

Si restringimos la variable t al conjunto de las tuplas de R, es decir, si DOM(t) = R, la última fórmula anterior se escribiría:

$\exists t(t_1 = 8)$

Cuantificador universal

Lo representaremos como: $\forall$t (condición).

Esto se lee: «Para todas las tuplas t se cumple la condición.»

La fórmula

$\forall$t (condición)

es verdadera si para todos los valores de la variable t se cumple la condición, y falsa en caso contrario.

Ejemplos:

La fórmula

$\forall t((\neg R(t)) \vee ((R(t)) \wedge (t_1 = 8)))$

es verdadera si todas las tuplas de R empiezan por 8.

Restringiendo t sólo a R, la fórmula anterior se escribiría:

Para DOM(t) = R:

$\forall t(t_1 = 8)$

Expresiones

Una expresión tiene la forma:

$(t_{(n} \mid F(t))$

donde F es una fórmula.

Significa: el conjunto de todas las tuplas t de grado n tales que la fórmula F(t) tome el valor verdadero. Puede omitirse el grado si está claro en el contexto.

La fórmula F(t) se construye combinando condiciones compuestas y cuantificadores.

Dentro de F(t), la única variable que puede estar no cuantificada es t. Es decir, podemos tener:

$(t \mid F(t, u, v, \ldots))$

donde t es la única que puede estar no cuantificada; las otras, u, v, ..., están cuantificadas, es decir, aparecen con ∃ o ∀.

Ejemplos de expresiones:

1) Formar una relación cogiendo la primera columna de R y la segunda de S:

$(t_{(2} \mid \exists r(R(r) \wedge t_1 = r_1) \wedge$
$\exists s(S(s) \wedge t_2 = s_2)$

Si la variable r toma valores sólo en R y la s en S, la expresión anterior se escribiría:

Para (DOM(r) = R) y (DOM(s) = S):

$(t_{(2} \mid \exists r(t_1 = r_1) \wedge \exists s(t_2 = s_2))$

2) Formar una tupla de grado 1 con el mínimo valor del primer atributo de R:

Para (DOM(r) = R) y (DOM(s) = R):

$(t_{(1} \mid \exists r(t_1 = r_1) \wedge \forall s(t_1 <= s_1))$

3) Extraer todas las tuplas de R si R está contenida en S. Si no, no extraer ninguna:

Para (DOM(r) = R):

$(t \mid R(t) \wedge \forall r(S(r)))$

4) Extraer todas las tuplas de R si hay alguna que empiece por 1 y otra que empiece por 2. Si no, no extraer ninguna:

Para $(DOM(r)=R)$ y $(DOM(s)=R)$:

$(t \mid R(t) \wedge \exists r(r_1=1) \wedge \exists s(s_1=2))$

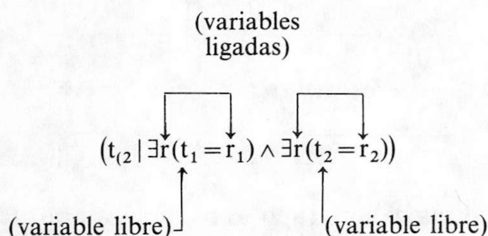
Variables libres y ligadas

En la escritura de fórmulas de cálculo se sigue el convenio de permitir que una misma variable pueda tomar valores diferentes, dentro de una misma fórmula. Cuando una variable está cuantificada, todas las ocurrencias de la misma que representan el mismo valor se llaman variables ligadas. Es un convenio análogo al uso de variables locales en programación modular. Las variables no ligadas se llaman libres.

Ejemplos:

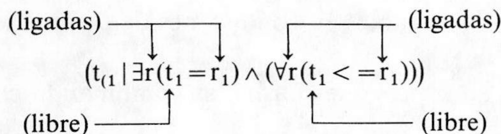
Con este convenio, los ejemplos de expresiones del apartado anterior se podrían haber escrito así también:

1) Para $(DOM(r)=R)$:

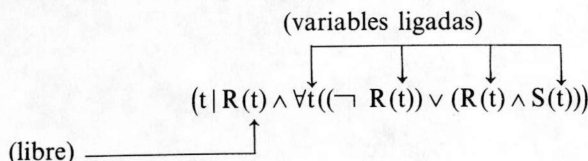


Se puede observar aquí que la variable r se usa dos veces, pudiendo tomar valores distintos cada vez. Es decir, posiblemente la tupla r que tiene $t_1=r_1$ no es la misma tupla r que tiene $t_2=r_2$.

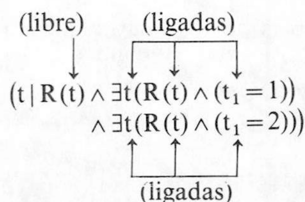
2) Para $(DOM(r)=R)$:



3)



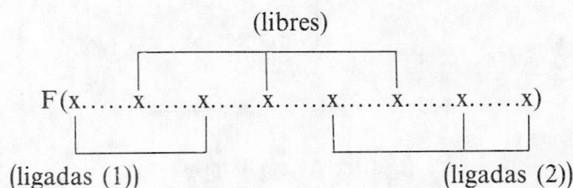
4)



Resumiendo, en toda fórmula F puede haber varias ocurrencias de una variable x. Cada ocurrencia puede ser libre o ligada, de forma que podemos clasificar sus ocurrencias en varios grupos:

- El grupo de ocurrencias libres.
- El primer grupo de ocurrencias ligadas entre sí.
- El segundo grupo de ocurrencias ligadas entre sí.
- Etc.

Podríamos tener, por ejemplo, una situación como la indicada en la siguiente figura:



Todas las ocurrencias de un grupo se reemplazan por la misma tupla cuando se quiere comprobar si una tupla verifica o no la fórmula F. En cambio, ocurrencias de distintos grupos pueden reemplazarse por diferentes tuplas.

Solamente puede haber un grupo de variables libres en F, y debe corresponder con la variable inicial, t, de la expresión

$(t \mid F(t, \ldots))$

Las ocurrencias de variables en condiciones de comparación o de pertenencia son libres en principio.

Las ocurrencias en expresiones formadas combinando condiciones con $\wedge$, $\vee$ y $\neg$ también son libres.

Las ocurrencias con cuantificadores son ligadas.

Así, si en F(t, u, v, ...) todas las ocurrencias de t, u, v, ..., son libres, al formar

$\exists t(F(t, u, v, \ldots))$, o
$\forall t(F(t, u, v, \ldots))$,

se ligan todas las ocurrencias de t entre sí.

Si en F(t, u, v, ...), t tiene ocurrencias libres y ligadas, en

$\exists t(F(t, u, v, \ldots))$, o en
$\exists t(F(t, u, v, \ldots))$,

todas las ocurrencias libres de t en F pasan a estar ligadas entre sí y con la ocurrencia de t que sigue al cuantificador.

CALCULO RESTRINGIDO DE TUPLAS

Con las definiciones dadas hasta aquí, sería válido escribir la sentencia

$(t_{(r} \mid \neg R(t))$

Esto expresa el conjunto de todas las tuplas de grado r pertenecientes al conjunto **TUP** que no están en R(t). Este conjunto puede ser infinito, por lo que puede ser imposible evaluar esta expresión.

Denominaremos fórmulas sanas a las que pueden evaluarse con un proceso o algoritmo finito.

Las fórmulas insanas pueden transformarse en otras sanas (no equivalentes) restringiendo los dominios de variación de algunas de sus variables a subconjuntos finitos de **TUP**.

Ejemplo:

Tomemos la fórmula insana anterior:

$(t_{(r} \mid \neg R(t))$

(Todas las tuplas de grado r de **TUP** que no están en R(t)).

Podemos transformarla en la siguiente fórmula sana (no equivalente):

Para DOM(t) = S:

$(t \mid \neg R(t))$

(Significado: Todas las tuplas de S que no están en R).

Tal como se definió previamente el álgebra, no se presentan en ella estos problemas. Puesto que los operandos de las expresiones algebraicas son finitos siempre (constantes o relaciones), los resultados de aplicarles los operadores primitivos también lo serán (recuérdese que en álgebra no se incluyó la operación de complementar). Dicho de otra forma, todas las expresiones algebraicas son sanas. Por tanto, toda expresión de cálculo equivalente a otra de álgebra, se podrá evaluar en un proceso finito, es decir, es sana.

Denominaremos cálculo restringido de tuplas a todas las expresiones de cálculo de tuplas que tienen alguna expresión algebraica equivalente. Según esta definición, todas las expresiones del cálculo restringido son sanas, y además, el álgebra es tan expresiva como el cálculo restringido. Recíprocamente, toda

expresión de álgebra puede expresarse en cálculo restringido, como veremos en el apartado siguiente, por lo que ambos son equipotentes.

POTENCIA EXPRESIVA DEL CALCULO RESTRINGIDO DE TUPLAS

Para toda expresión algebraica existe una fórmula de cálculo equivalente, por lo que el cálculo restringido y el álgebra tienen la misma potencia expresiva.

Veamos cómo expresar los operadores primitivos del álgebra en fórmulas de cálculo.

1) *Unión.*

Sea $E = R \cup S$. Podemos expresar E en cálculo como sigue:

$$E = (t \mid R(t) \vee S(t))$$

2) *Diferencia.*

$$E = R - S$$
$$E = (t \mid R(t) \wedge (\neg S(t)))$$

3) *Producto cartesiano.*

$$E = R \times S$$

Para R de grado r, S de grado s, DOM(u) = R y DOM(v) = S:

$$E = (t_{(r+s} \mid (\exists u)\ (\exists v)$$
$$((t_1 = u_1) \wedge (t_2 = u_2) \wedge \ldots (t_r = u_r)$$
$$\wedge\ (t_{r+1} = v_1) \wedge (t_{r+2} = v_2) \wedge \ldots (t_{r+s} = v_s)))$$

4) *Proyección.*

$$E = \mathbf{P}(A_1, A_2, A_3, \ldots)\ (R)$$
$$E = (t \mid \exists u (R(u) \wedge (t_1 = u_1) \wedge (t_2 = u_2) \wedge \ldots)$$

5) *Selección.*

$$E = \mathbf{S}(F)\ (R)$$
$$E = (t \mid R(t) \wedge (F))$$

Por ejemplo

$$(\mathbf{S}(t_1 = t_2)\ (R)) = (t \mid R(t) \wedge (t_1 = t_2))$$

Como ejemplo, y para completar la lista anterior, veamos las operaciones de intersección y cociente:

$$E1 = (R \cap S)$$
$$E1 = (t \mid R(t) \wedge S(t))$$
$$E2 = (R_{(m+n}/S_{(n)}$$

Para DOM(r) = R y DOM(s) = S:

$$E2 = \big(t_{(m} \mid \forall s(\exists r((r(1) = t(1)) \wedge$$
$$\big(r(2) = t(2)) \wedge \ldots (r(m) = t(m)\big) \wedge$$
$$\big(r(m+1) = s(1)\big) \wedge (r(m+2) = s(2)) \ldots \wedge$$
$$\big(r(m+n) = s(n)\big)$$

Si consideramos una expresión algebraica, E, sin operadores, sólo puede tratarse de una relación constante o variable. En ambos casos es expresable en cálculo. Como cualquier expresión algebraica se puede construir recurrentemente a partir de constantes y variables combinadas con operadores primitivos, se deduce que es expresable en cálculo.

LENGUAJES PROCEDIMENTALES («PROCEDURAL»)

El álgebra es un lenguaje procedimental. Esto significa que para indicar el resultado que queremos obtener, enunciamos cómo obtenerlo. Es decir, qué operaciones y en qué secuencia hay que realizarlas para llegar al resultado deseado.

El cálculo no es procedimental. En cálculo se indica directamente qué resultados deseamos, sin indicar cómo llegar a él.

Parece, por tanto, en principio, que el cálculo será un lenguaje preferible al álgebra, puesto que el usuario sólo tiene que definir el resultado final que desea obtener, siendo responsabilidad del sistema gestor de la base de datos (SGBD) la definición de la secuencia de operaciones elementales necesarias para evaluar la petición.

Sin embargo, como ambos lenguajes son equipotentes, esta discusión es irrelevante, puesto que el SGBD puede transformar una sentencia expresada en uno de ellos al otro. Por otra parte, cuando el usuario expresa una petición, tanto en álgebra como en cálculo, el SGBD deberá hallar el procedimiento óptimo para responderla. Esto hace que, en la práctica, aunque el usuario se exprese en álgebra, la secuencia de procesos pueda ser diferente a la indicada en la petición, por lo que, de hecho, el álgebra es procedimental cara al usuario, pero no dentro del sistema.

En definitiva, también el álgebra es, desde este punto de vista, simplemente un medio de expresar lo que se quiere conseguir, y no el cómo conseguirlo (se expresa el qué mediante el cómo).

Desde un punto de vista psicológico, se ha detectado en algunas experiencias que el uso de un lenguaje procedimental puede ser más sencillo, pues permite descomponer una petición en varios pasos más fácilmente que un lenguaje no procedimental. De todas formas, esto no parece concluyente, pues depende de las características psicológicas de cada persona.

En definitiva, la discusión comparativa entre los méritos de los lenguajes del tipo álgebra o del tipo cálculo es irrelevante desde el punto de vista del SGBD.

Algunas consultas son más fácilmente expresables en cálculo que en álgebra, y otras al revés.

Ejemplos:

1) Hallar el valor mínimo del primer atributo de R.

   Cálculo:

   Para DOM(r) = R y DOM(s) = R,

   $(t_{(1} \mid \exists r(t_1 = r_1) \wedge \forall s(t_1 <= s_1))$

   Algebra:

   Sean A_1, A_2, ..., los atributos de R. Sea T una relación igual a la R, con atributos B_1, B_2, ... Es decir, que $\mathbf{P}(A_1)$ (R) = $\mathbf{P}(B_1)$ (T). El valor mínimo de A_1 viene dado por:

   $[\mathbf{S}(A_1 <= B_1) ((\mathbf{P}(A_1) (R)) \mathbf{x} (\mathbf{P}(B_1) (T)))]$
   $/(\mathrm{P}(A_1) (R))$

2) Sean R(A, B) y S(B, C) dos relaciones binarias con atributos A, B y C. Se llama composición de estas relaciones en teoría de conjuntos a la relación binaria obtenida al concatenar todos los valores de A que se corresponden con los de C a través de valores comunes de B. Hallemos esta relación en álgebra y cálculo.

   Cálculo:

   $(t \mid (\exists r) (\exists s) (R(r) \wedge S(s) \wedge$
   $(r_2 = s_1) \wedge (t_1 = r_1) \wedge (t_2 = s_2)))$

   Algebra:

   $\mathbf{P}(A, C) (R * S)$

LENGUAJES RELACIONALMENTE COMPLETOS

Tanto el álgebra como el cálculo restringido de tuplas (y el de dominios, que mencionaremos más adelante), son equipotentes. Es decir, cualquier consulta expresada en uno de ellos puede expresarse en los otros.

Parece natural entonces considerar a estos lenguajes como patrón o medida para estudiar la potencia expresiva de otros.

Se dice que un lenguaje es relacionalmente completo si tiene, al menos, la potencia expresiva del álgebra relacional. Es decir, si para toda expresión de álgebra existe alguna sentencia equivalente en ese lenguaje.

Dicho de otra forma, el álgebra y el cálculo relacionales son lenguajes relacionalmente completos. Esto no quiere decir que nos permitan expresar cualquier tipo de consulta imaginable.

En álgebra no existen bucles iterativos ni manipulaciones aritméticas. Tampoco existe el concepto de orden o clasificación entre las tuplas de una relación resultante de otras.

DEFINICION FORMALIZADA DEL CALCULO DE TUPLAS

- Las expresiones son de la forma $(t_{(n} \mid F(t))$, donde t es una variable sobre las tuplas de **TUP** de grado n y F es una fórmula.
- F(t) es una fórmula construida con átomos y operadores. Los átomos pueden ser de la forma:
  1) $(R(s))$, donde R es una relación y s una variable.
  2) $(s(i)\mathbf{C}u(j))$, donde s y u son variables, **C** es un operador de comparación, i y j son identificadores de componentes de las tuplas s y u.
  3) $(s(i)\mathbf{C}a)$ o $(a\mathbf{C}s(i))$, donde s, i y **C** son como antes, y a es una constante.
- Todo átomo es una fórmula. En ella todas las ocurrencias de variables son libres.
- Si F1 y F2 son fórmulas, también lo son $(F1 \wedge F2)$, $(F1 \vee F2)$, $(\neg F1)$. En estas últimas, las ocurrencias de variables son libres o ligadas, según lo sean en F1 o en F2.
- Si F es una fórmula, también lo es $(\exists s(F))$. Las ocurrencias de s que sean libres en F, están ligadas en $(\exists s(F))$ a la s que sigue al cuantificador $\exists$. Otras ocurrencias de variables en F son ligadas o libres en $(\exists s(F))$ según sean en F.
- Si F es una fórmula, también lo es $(\forall s(F))$. Las ocurrencias libres y ligadas se definen como en el caso anterior.
- Pueden usarse paréntesis.
- Ninguna otra cosa es una fórmula.
- En una expresión $(t \mid F(t))$, sólo t puede tener ocurrencias libres en la fórmula F.

INTRODUCCION INFORMAL AL CALCULO DE DOMINIOS

Se define igual que el cálculo de tuplas, pero manejando variables cuyos campos de valores posibles son los elementos de los dominios.

Ejemplos:

1) Extraer todas las tuplas de una relación binaria R, si R contiene dos o más tuplas. Si no, no extraer ninguna.

   Cálculo de tuplas:

   $(t_{(2} \mid \exists u(R(t) \wedge R(u) \wedge ((t_1 \neg = u_1) \vee (t_2 \neg = u_2)))$

   Cálculo de dominios (x, y, z, w, son variables de dominio):

   $(xy \mid (\exists z)\ (\exists w)\ (R(xy) \wedge R(zw) \wedge ((x \neg = z) \vee (y \neg = w)))$

2) Extraer todas las tuplas de $R_{(2}$ que empiecen por 1 si hay alguna que empiece por 2. Si no, no extraer ninguna.

Cálculo de tuplas:

$(t_{(2} \mid R(t) \wedge (t_1 = 1) \wedge (\exists u(R(u) \wedge (u_1 = 2))))$

Cálculo de dominios:

$(xy \mid R(xy) \wedge (x = 1) \wedge (\exists z((\exists w(R(zw))) \wedge (z = 2))))$

3) Extraer todas las tuplas de $R_{(2}$ que empiecen por 1 si no hay ninguna que empiece por 2. Si no, no extraer ninguna.

Cálculo de tuplas:

Para $DOM(s) = R$:

$(t \mid R(t) \wedge (t_1 = 1) \wedge (\forall s(s_1 \neg = 2)))$

Cálculo de dominios:

Para $DOM(z) =$ (dominio del segundo atributo de R):

$(xy \mid R(xy) \wedge (x = 1) \wedge (\forall z(\neg R(2z))))$

O también:

$(xy \mid R(xy) \wedge (x = 1) \wedge \neg \exists z(R(2z)))$

En el cálculo de dominios hay que indicar sobre qué dominio toma valores cada variable si no se deduce claramente del contexto.

El cálculo de dominios restringido se define análogamente al de tuplas, y es equipotente con él y con el álgebra.

EJERCICIOS PROPUESTOS

1) *Responder en cálculo a las preguntas del ejercicio 2 del capítulo anterior.*

2) *Responder en cálculo a las preguntas del ejercicio 3 del capítulo anterior.*

3) *Responder en cálculo a las preguntas del ejercicio 4 del capítulo anterior.*

Normalización

4

CONDICIONES DE INTEGRIDAD

Sea **B** un esquema de base de datos, es decir, un conjunto de esquemas de relaciones, $\mathbf{B}=(R_1, R_2, R_3, \ldots, R_m)$.

El contenido de esta base de datos varía con el tiempo mediante borrado, añadido o modificación de sus tuplas.

De acuerdo con el significado de las relaciones de **B**, no todas las tuplas posibles construidas sobre los dominios de un esquema de relación R_i pueden estar incluidas en una extensión de R_i. Normalmente, sólo las tuplas que cumplan ciertas condiciones o propiedades, podrán ser elementos válidos de R_i, según lo que R_i signifique.

Llamaremos **CINT** al conjunto de las condiciones que deben cumplir las tuplas de **B** a lo largo del tiempo, en función del significado de los distintos esquemas de relación de **B**. Estas condiciones se llaman condiciones de integridad, y en ellas se reflejan propiedades que son invariantes al transcurrir el tiempo. (En la fugacidad de este mundo pocas cosas hay eternas, incluyendo las condiciones de integridad. Al decir invariantes con el tiempo nos referimos a un lapso de tiempo arbitrariamente largo, de acuerdo con las necesidades de uso de los datos.)

Las condiciones de **CINT** pueden referirse a tuplas o atributos de una relación (condiciones intrarrelación) o de varias (condiciones interrelaciones). Expresan restricciones semánticas, es decir, dependen del significado de las relaciones y sus atributos.

Podemos distinguir dos tipos de condiciones de integridad:

1) *Condiciones de estado:* Para comprobar si una extensión determinada las verifica (o sea, si es un estado válido), basta mirar el contenido de una, varias o todas las relaciones del esquema. Es decir, que basta tener en cuenta las extensiones actuales de una o varias relaciones del esquema para saber si es un estado válido o no.

2) *Condiciones de transición:* Enuncian si, a partir de un estado válido, el obtenido aplicando un cierto cambio (insertar, actualizar o borrar n tuplas) es también válido o no. Es decir, para saber si el nuevo estado es válido, hace falta información sobre las extensiones actuales y las previas (por ejemplo si ha de cumplirse que toda actualización de una cierta fecha ha de ser con un valor posterior al que sustituye).

Aunque las condiciones de integridad son consecuencia del significado de los datos, se expresan en función de los valores de éstos exclusivamente, por lo que, para comprobar si se cumplen, no es necesario conocer el significado de los datos, sino que basta con mirar el contenido de las relaciones (actual o pasado).

Ejemplo:

Sea el esquema:

PED (NUMPEDIDO, NUMPROVEEDOR)
LIPE (NUMLINEA, NUMPEDIDO, NUMARTICULO, CANTIDAD)

En este esquema, cada tupla de PED representa un pedido. Por tanto, para saber si es válido insertar la tupla (100, 1000), un criterio es si existe un pedido de número 100 al proveedor 1000. Si es así, hay que incluir esta tupla en PED, y en caso contrario, no. Evidentemente, este criterio de validez para la operación de insertar es ajeno al contenido actual o pasado de PED, por lo que no es una condición de integridad.

Sí sería en cambio una condición de integridad la siguiente: todo valor de NUMPEDIDO en LIPE debe existir en NUMPEDIDO de PED. Para comprobar si esta condición se cumple o no, basta considerar los valores contenidos en PED y LIPE, independientemente de qué representan estas relaciones.

Ejemplos de condiciones de integridad

1) Para toda tupla de un esquema de relación dado, el valor de un atributo debe ser inferior a otro. Ejemplo:

EMP(NRO, FECHANAC, FECHAINGRESO)

Evidentemente, para todo empleado la fecha de nacimiento deberá ser anterior a la de su ingreso en la empresa. Esta condición se podría expresar en cálculo:

Para DOM(t) = EMP:

$\forall t(\text{FECHANAC} < \text{FECHAINGRESO})$

2) Para un esquema de relación dado, R, los valores de un conjunto, A, de sus atributos no pueden repetirse (es decir, que A es una superclave de R).

En cálculo:

Para DOM(t) = R, DOM(u) = R:

$(\forall t)\ (\forall u)\ ((t = u) \vee (t(A) \neg = u(A)))$

3) Para dos esquemas de relación dados, R y S, el valor mínimo de un cierto atributo B, de S, debe ser siempre mayor que el mínimo de otro atributo, A, de R.

En cálculo:

Para DOM(t) = R, DOM(u) = S:

$(\exists t(\forall u(t(A) <= u(B))))$

Ejemplo:

COCHES (MARCA, MODELO, FECHALANZAMIENTO)
EMP-COCHE (NROEMP, FECHACOMPRA)

Evidentemente, ningún empleado puede comprar un coche antes de que haya alguno disponible en el mercado. Es decir, el mínimo valor de FECHACOMPRA debe ser mayor que el mínimo de FECHALANZAMIENTO.

4) Para dos esquemas dados, todo valor de un atributo, A, de uno de ellos, debe existir en el atributo B del otro.

En cálculo:

Para DOM(t) = R, DOM(u) = S:
$(\forall t(\exists u(t(A) = u(B))))$

Ejemplo:

ARTICULO (NUMART, DESCRIPCION)
PEDIDO (NUMART, PROVEEDOR, CANTIDAD)

Todo artículo pedido (atributo PEDIDO.NUMART) debe existir en la tabla de artículos (atributo ARTICULO.NUMART).

Consecuencia: no se puede insertar una nueva fila en la tabla PEDIDO si no existe previamente una fila en la tabla ARTICULO con el mismo valor de NUMART; no se puede borrar una fila de ARTICULO si hay pedidos de él en PEDIDO.

DISEÑOS EQUIVALENTES

Dado un esquema de base de datos **B**, se puede plantear la pregunta de si es posible encontrar otro, **B**′, equivalente a él y cuál de ellos es preferible.

El nuevo diseño, **B**′, deberá contener al menos la misma información que el viejo, **B**.

Diremos que **B**′ contiene al menos tanta información como **B**, si existe un algoritmo que nos permite obtener una extensión cualquiera de **B** a partir de una de **B**′.

Ejemplo 1:

B:

PED (ART, PROVD, CANT, PRECIO)

B′:

PED (ART, PROVD, CANT)
INVENTARIO (ART, PRECIO, CANTIDAD)

Algoritmo (expresado en álgebra) para hallar PED de **B** en función de las relaciones de **B**′:

(**B**.PED) = **P**(ART,PROVD,CANT,PRECIO) ((**B**′.PED) * (**B**′.INVENTARIO))

En este ejemplo, **B**′, contiene más información que **B**:

- Hay atributos en **B**′ que no están en **B** (el atributo CANTIDAD, que expresa la cantidad que hay en existencias en almacén de un determinado artículo).
- Puede haber tuplas en **B**′ que no puedan existir en **B**. Por ejemplo, en INVENTARIO puede haber artículos que no estén pedidos, y que por tanto no figurarán en PED.

Ejemplo 2:

B:

COCHES (MARCA, MODELO, FECHALANZ)

B′:

COCHESMODERNOS (MARCA, MODELO, FECHALANZ)
COCHESVIEJOS (MARCA, MODELO, FECHALANZ)

Algoritmo (expresado en álgebra):

COCHESMODERNOS = **S** (FECHALANZ > = 1980) (COCHES)
COCHESVIEJOS = **S** (FECHALANZ) < 1980) (COCHES)
(COCHES) = (COCHESMODERNOS) $\cup$ (COCHESVIEJOS)

Evidentemente, **B**′ y **B** contienen la misma información.

Para decidir si es preferible el esquema o diseño representado por **B**′ o por **B**, hay que tener en cuenta las condiciones de integridad. Es posible que las condiciones **CINT** aplicables a **B**, no lo sean a **B**′. En general, las condiciones **CINT** aplicables a **B** se tranformarán en otras distintas **CINT**′ aplicables a **B**′.

Veamos esto en los ejemplos siguientes.

Ejemplo 3:

En el ejemplo 1 anterior teníamos las siguientes condiciones de integridad:

CINT:

A cada artículo corresponde un precio. Para DOM(t) = PED, DOM(u) = PED:

$(\forall t)\ (\forall u)\ ((t(ART) \neg = u(ART)) \vee ((t(ART) = u(ART)) \wedge (t(PRECIO) = u(PRECIO))$

Además, (ART,PROVD) es una superclave (es decir, toma valores no repetidos).

CINT′:

Para toda fila de PED debe existir una en INVENTARIO con el mismo número de artículo. Es decir:

Para DOM(t) = B′.PED, DOM(u) = INVENTARIO:

$(\forall t(\exists u(t(ART) = u(ART))))$

Además, ART es superclave de INVENTARIO, y (ART,PROVD) de PED.

Ejemplo 4:

En el ejemplo 2 anterior tenemos las condiciones siguientes:

CINT: No hay restricciones (**CINT** = ∅).
CINT':

Para toda fila de COCHESMODERNOS, debe ser la fecha de lanzamiento posterior a 1980. Para las filas de COCHESVIEJOS deberá ser anterior.

En cálculo para DOM(t) = COCHESMODERNOS y DOM(u) = COCHESVIEJOS:

(∀t(t(FECHALANZ) > = 1980)),
(∀u(u(FECHALANZ) < 1980))

En definitiva, dados dos esquemas de diseño, (**B**, **CINT**) y (**B'**, **CINT'**), tales que **B'** contenga al menos tanta información como **B**, debemos decidir cuál es preferible. Esto depende, entre otras cosas, del uso que se haga de la información. Si predominan las actualizaciones sobre las extracciones de datos (consultas), será más eficiente el esquema cuyas condiciones de integridad sean más fácilmente verificables (menos accesos a disco, por ejemplo) en el momento de actualizarlo (o sea, al añadir, borrar o modificar alguna tupla). Así, en el ejemplo 4 anterior teníamos (**CINT** = ∅), por lo que se puede actualizar el esquema **B** sin ninguna restricción, al contrario que el **B'**, donde hay que comprobar cada vez que se actualice si las condiciones **CINT'** se cumplen. Por el contrario, para consultas por fecha sería más eficiente, en principio, el esquema **B'** (menos tuplas que investigar).

REVERSIBILIDAD POR YUNCION

Si descomponemos un esquema R en otros mediante proyecciones, ¿se obtiene un diseño equivalente? Analicemos esta cuestión.

Dado un esquema de relación, R, no siempre es posible descomponerlo en otros mediante proyecciones de manera que al ayuntar éstos, se reproduzca R. En caso de serlo, decimos que esta descomposición de R es reversible por yunción («lossless join decomposition»).

Ante todo precisemos la forma de descomponer. Supongamos que descomponemos el esquema R mediante dos proyecciones en S y T. Sea U el conjunto de todos los atributos de R, y sean A, B y C subconjuntos propios de U disjuntos y que cubren totalmente a U. Es decir:

$A\neg = \emptyset$; $B\neg = \emptyset$; $C\neg = \emptyset$;
$A \cap B = \emptyset$; $A \cap C = \emptyset$; $B \cap C = \emptyset$;
$A \subset U$; $B \subset U$; $C \subset U$;
$A \cup B \cup C = U$

Entonces, proyectamos R de la forma siguiente:

S = **P**(A, B) (R);
T = **P**(A, C) (R);

Vamos a analizar si esta descomposición es reversible, es decir, si R = S * T.

Sea ⟨a, b, c⟩ una tupla cualquiera de R(A, B, C). Entonces, ⟨a, b⟩ estará en S y ⟨a, c⟩ estará en T. Por tanto, ⟨a, b, c⟩ estará en la yunción de S y T. Es decir, R está contenida en la yunción de S y T:

$R \subseteq (S * T)$

En general, al ayuntar S y T, se obtienen todas las tuplas de R y algunas más que no están en R.

En efecto, si ⟨a, b, c⟩ no está en R, pero sí están en R ⟨a, b, d⟩ y ⟨a, e, c⟩, tendremos que como ⟨a, b⟩ está en S y ⟨a, c⟩ en T, ⟨a, b, c⟩ estará en la yunción de S y T. Es decir, en la yunción de S y T aparece la tupla ⟨a, b, c⟩ que no está en R. Por tanto, en general, $(R \subset (S * T))$. Veamos qué condiciones debe cumplir R para que sea $(R = (S * T))$:

La condición necesaria y suficiente para que la descomposición anterior sea reversible por yunción es que si existen dos tuplas en R con el mismo valor en A, también existan las que resulten de intercambiar entre ellas los valores de B.

Ejemplo 1:

Si A es una superclave de R (o sea, toma valores únicos en R), se cumple la condición anterior, y por tanto la descomposición es reversible:

R	A	B	C
	a	m	1
	b	m	1
	c	n	1
	d	m	2

S	A	B
	a	m
	b	m
	c	n
	d	m

T	A	C
	a	1
	b	1
	c	1
	d	2

Ejemplo 2:

Si en las tuplas donde se repite A, también se repite B, también se cumple la condición anterior, y por tanto la descomposición es reversible. Este tipo de condición es muy importante, y recibe un nombre específico: se dice que B *depende funcionalmente* de A, y se representa $(A \rightarrow B)$. También diremos que *«A define funcionalmente a B»:*

R	A	B	C
	a	m	1
	a	m	2
	a	m	3
	b	n	1
	b	n	3
	c	m	1

S	A	B
	a	m
	b	n
	c	m

T	A	C
	a	1
	a	2
	a	3
	b	1
	b	3
	c	1

Vemos que R = (S * T).

Ejemplo 3:

Un ejemplo en el caso general:

R	A	B	C
	a	m	1
	a	m	2
	a	n	1
	a	n	2
	b	m	3
	b	n	3

S	A	B
	a	m
	a	n
	b	m
	b	n

T	A	C
	a	1
	a	2
	b	3

Vemos aquí que en todas las tuplas en que se repite A, intercambiando los valores de B, se obtienen otras tuplas que también están en R. Este tipo de condición también tiene la importancia suficiente como para designarla con un nombre específico: se dice que hay *una dependencia plural entre A y B,* o también que B depende pluralmente de A, o que A pluridefine a B, y se representa $(A \twoheadrightarrow B)$.

En definitiva, la condición necesaria y suficiente para que la descomposición sea reversible se representa en general como $(A \twoheadrightarrow B)$.

Por otra parte, también hemos visto que si se verifica que $(A \rightarrow B)$ también se verifica $(A \twoheadrightarrow B)$.

Además, como A, B y C son disjuntos y cubren totalmente a U, la condición $(A \twoheadrightarrow B)$ es equivalente a $(A \twoheadrightarrow C)$. En efecto, por simetría, lo mismo da decir que al repetirse A se obtienen tuplas de R si se intercambian los valores de B que si se intercambian los valores de C.

Vamos a ver que $(A \twoheadrightarrow B)$ es condición necesaria y suficiente para que la descomposición sea reversible. Veamos primero que es suficiente, porque si se cumple, la descomposición es reversible. En efecto, supongamos que tomamos dos tuplas que empiecen por a en S y T. Por ejemplo: ⟨aA⟩ en S y ⟨a1⟩ en T. Al ayuntar S y T se obtiene que ⟨aA1⟩ está en $(S * T)$. Por otra parte, el hecho de que ⟨aA⟩ esté en S, quiere decir que hay en R una tupla de la forma ⟨aAx⟩. Análogamente, por estar ⟨a1⟩ en T, en R existe una tupla de la forma ⟨a y 1⟩. Intercambiando entre ellas los valores de B, obtendremos las tuplas ⟨a y x⟩ y ⟨aA1⟩, que también existirán en R. Es decir, si se cumple la condición $(A \twoheadrightarrow B)$, toda tupla de $(S * T)$ está en R. Por tanto, la descomposición es reversible.

Veamos ahora el recíproco: que si la descomposición es reversible, se cumple $(A \twoheadrightarrow B)$.

En efecto, supongamos que la descomposición es reversible. Esto quiere decir que toda tupla de $(S * T)$ está en R.

Supongamos que tomamos dos tuplas de S y T que empiecen por a, tales como ⟨aA⟩ de S y ⟨a1⟩ de T. Esto quiere decir que existen en R dos tuplas de la forma ⟨aAx⟩ y ⟨a y 1⟩. Pero esto implica también que ⟨ay⟩ está en S y ⟨ax⟩ en T. Por tanto, $(S * T)$ contiene las tuplas ⟨aA1⟩, ⟨aAx⟩, ⟨a y 1⟩ y ⟨a y x⟩. Como la descomposición es reversible, estas tuplas también están en R. Luego se cumple $(A \twoheadrightarrow B)$.

La definición dada para la condición $A \twoheadrightarrow B$ es extensible y válida para el caso en que A y B no sean disjuntos. Así lo supondremos de ahora en adelante mientras no se diga lo contrario.

En definitiva, supongamos que tenemos un esquema de diseño,(**B**, **CINT**), formado por un esquema de base de datos, **B**, y un conjunto de condiciones de integridad, **CINT**, donde **B** contiene un solo esquema de relación, R, y **CINT** una sola condición expresada por ($A \twoheadrightarrow B$), donde A y B son subconjuntos de U (los atributos de R) propios y disjuntos. Como ($A \twoheadrightarrow B$), podemos descomponer R en (S = **P**(A, B) (R)) y (T = **P**(A, C) (R)) de forma reversible por yunción, como ya hemos visto. Supongamos entonces que reemplazamos el esquema de diseño de partida (**B**, **CINT**) por otro (**B′**, **CINT′**), donde **B′** = (S, T). Veamos si este nuevo esquema de diseño reemplaza válidamente al primero.

Evidentemente, cualesquiera que sean las tuplas de S y T, se puede recomponer un estado válido de R. Por tanto, (**CINT′** = $\emptyset$) (no hay condiciones). Esto quiere decir que el nuevo esquema es más fácil de actualizar que el original.

Por otra parte, el nuevo esquema de diseño puede contener más información que el original, pues si incluimos en S una tupla que empiece por un valor de A que no exista en T, esta tupla no se reflejará en R.

Ejemplo:

S	A	B
	a	A
	b	A
	c	B

T	A	C
	a	1
	a	2
	b	1

(S * T)	A	B	C
	a	A	1
	a	A	2
	b	A	1

(La tupla ⟨cB⟩ de S no aparece en la yunción.)

En resumen, el nuevo esquema de diseño es válido, pues contiene toda la información del original (contiene más), y es más sencillo de mantener, pues no es necesario comprobar nada al actualizar tuplas. Cualquier tupla es válida en S y T. Los esquemas son:

- Primer esquema de diseño:

  B = (R); **CINT** = ($A \twoheadrightarrow B$);

- Segundo esquema de diseño:

  B′ = (S, T); **CINT′** = $\emptyset$;

  donde S = **P**(A, B) (R), y T = **P**(A, C) (R).

- Algoritmo para reconstruir una extensión de **B** a partir de **B′**:

  R = (S * T)

Supongamos ahora que **CINT** en vez de contener una sola condición, ($A \twoheadrightarrow B$), contiene varias, y además de cualquier tipo (dependencias plurales, dependencias

funcionales, condiciones aritméticas, etc.). Ahora ya **CINT′** no sería probablemente vacío, y habría que analizar la validez del esquema **B′** obtenido mediante las proyecciones de R.

Para estudiar este problema, vamos a simplificarlo restringiendo en principio las condiciones de **CINT** a sólo dos tipos: dependencias plurales y funcionales, y de momento vamos a considerar sólo estas últimas.

DEPENDENCIAS FUNCIONALES

Hemos mencionado ya este tipo de condiciones.

Las dependencias funcionales son condiciones de integridad que se definen como sigue:

> **Sean A y B subconjuntos propios de U (U es el conjunto de los atributos de una relación R). Decimos que A define funcionalmente a B en R si para dos tuplas cualesquiera de R que tengan iguales valores en A, también son iguales los valores de B. Se representa (A→B).**

Las dependencias funcionales son condiciones de integridad entre atributos de una misma relación. No permiten expresar condiciones entre atributos de relaciones diferentes.

Por ahora vamos a considerar esquemas de diseño en los que todas las condiciones de integridad sean expresables como dependencias funcionales. Entonces designaremos con **DF** al conjunto de condiciones aplicables, con lo que un esquema de diseño se representará **(B, DF)**.

Ejemplo:

B:

PED (ART, PROV, CANT, PRECIO, CIUDAD, KM)

DF:

(ART, PROV)→(CANT, PRECIO, CIUDAD, KM)
ART→PRECIO
PROV→CIUDAD
CIUDAD→KM

PED	ART	PROV	CANT	PRECIO	CIUDAD	KM
	A1	P1	20	10	MAD	20
	A1	P2	20	10	VAL	400
	A1	P3	20	10	BNA	600
	A2	P1	30	20	MAD	20
	A2	P2	30	20	VAL	400
	A3	P1	40	30	MAD	20

La primera DF del ejemplo anterior indica que no puede haber dos tuplas con iguales valores en (ART, PROV), pues entonces las dos tuplas serían iguales,

lo que es imposible. Es decir, que (ART, PROV) es una superclave de la relación PED.

Es evidente además que de unas DFs pueden inferirse otras. En el ejemplo anterior de

PROV→CIUDAD
CIUDAD→KM

puede inferirse que, como a cada proveedor corresponde una ciudad, y a cada ciudad una distancia, también se puede decir que a cada proveedor corresponde una distancia, o sea: PROV→KM.

Además, como (ART, PROV) toma valores únicos, podemos decir que:

(ART, PROV)→CANT;
(ART, PROV)→PRECIO;
(ART, PROV)→CIUDAD;
(ART, PROV)→KM;

En definitiva, el conjunto **DF** dado es equivalente a:

(ART, PROV)→(CANT, PRECIO, CIUDAD, KM);
(ART, PROV)→CANT;
(ART, PROV)→PRECIO;
(ART, PROV)→CIUDAD;
(ART, PROV)→KM;
ART→PRECIO;
PROV→CIUDAD;
CIUDAD→KM;
PROV→KM;

Vemos por tanto que a partir de un conjunto de dependencias funcionales pueden inferirse otras. A continuación vamos a describir un sistema completo de reglas de inferencia aplicable a las dependencias funcionales.

SISTEMA DE INFERENCIA PARA DEPENDENCIAS FUNCIONALES

Diremos que una condición de integridad cualquiera se deduce o infiere de otras si se cumple en todas las extensiones posibles en que se cumplan éstas.

Sea R un esquema de relación, U el conjunto de sus atributos y X, Y, Z, W, subconjuntos propios de U. Entonces puede demostrarse que se verifican las propiedades o reglas de inferencia siguientes:

1) *Reflexividad:*

Si (Y⊆X), entonces se verifica que (X→Y).

2) *Aumentación:*

Si (X→Y), entonces se verifica que (XZ→YZ).

(Nota: XZ es una notación para indicar el conjunto formado por todos los atributos incluidos en X o en Z, es decir, es lo mismo que (X ∪ Z)).

3) *Transitividad:*

Si (X→Y) y (Y→Z), entonces se verifica que (X→Z).

Estas propiedades forman un sistema completo de reglas de inferencia. Es decir, todas las dependencias funcionales inferibles de un conjunto dado se pueden obtener aplicándolas a éste. Además, es sano. Es decir, todas las dependencias funcionales que se obtienen aplicando estas reglas son inferibles de las dadas. Este conjunto de reglas suele conocerse como axiomas de Armstrong. A partir de ellas pueden deducirse otras propiedades interesantes, como las siguientes:

- *Unión:*

  Si (X→Y) y (X→Z), entonces (X→YZ).

  En efecto, por aumentación: (X→XY), (XY→ZY).
  De aquí, por transitividad: (X→YZ), c.q.d.

- *Pseudotransitividad:*

  Si (X→Y) y (WY→Z), entonces (WX→Z).

  En efecto, por aumentación: (WX→WY). De aquí, por transitividad: (WX→Z), c.q.d.

- *Descomposición:*

  Si (X→Y) y (Z⊆Y), entonces (X→Z).

  En efecto, por reflexividad: (Y→Z). De aquí, por transitividad: (X→Z), c.q.d.

- De las propiedades de unión y descomposición anteriores se deduce la siguiente:

  Si A_1, A_2, A_3, ..., A_n, son los atributos de R, se cumple que ($X \rightarrow A_1 A_2 A_3 \ldots A_n$) si, y sólo sí, se verifica que ($X \rightarrow A_i$), para todo i(i=1, 2, 3, ..., n).

CLAUSURA

Dado un conjunto de dependencias funcionales, **DF**, llamaremos clausura de **DF**, y la representaremos por **DF+**, al conjunto de todas las dependencias

funcionales posibles que sean inferibles a partir de **DF**. Es decir, si aplicamos reiteradamente los axiomas de Armstrong a **DF** para inferir todas las dependencias funcionales que se pueda, todas ellas junto con **DF**, forman el conjunto, **DF+**, de la clausura de **DF**.

El conjunto **DF+** puede contener un gran número de elementos, por lo que puede ser prohibitivo ejecutar un algoritmo para construirlo. En cambio, sí existen algoritmos viables para determinar si una dependencia cualquiera dada, tal que (X→Y), está o no en **DF+**. Estos algoritmos nos permiten determinar si dos conjuntos dados de dependencias funcionales son equivalentes.

Dos conjuntos de dependencias funcionales, **DF1** y **DF2**, son equivalentes si, y sólo si, (**DF1+ = DF2+**). Es decir, son equivalentes si todas las dependencias funcionales inferibles de **DF1** también lo son a partir de **DF2**. O lo que es igual, si todas las dependencias de **DF1** están **DF2+**, y todas las de **DF2** están en **DF1+**. O lo que es igual, si todas las dependencias de **DF1** son inferibles a partir de **DF2**, y viceversa.

Ejemplo:

DF1:

(ART, PROV)→(CANT, PRECIO, CIUDAD, KM);
ART→PRECIO;
PROV→CIUDAD;
CIUDAD→KM;

DF2:

(ART, PROV)→CANT;
ART→PRECIO;
PROV→CIUDAD;
CIUDAD→KM;

DF1 y **DF2** son equivalentes.

RECONSIDERACIONES SOBRE LAS CLAVES

Ya hemos dicho que si X toma valores únicos, no repetidos, en R, X es una superclave de R. Vamos a reconsiderar este concepto.

Definición de superclave:

Sean A_1, A_2, A_3, ..., A_n, los atributos de R; X es una superclave de R si, y sólo si se verifica:

$$X \rightarrow (A_1 A_2 A_3 \ldots A_n)$$

Como ya se ha dicho anteriormente, si X es una superclave tal que ningún subconjunto propio de X sea también superclave, entonces se dice que X es una clave.

Definición de clave:

X es una clave si, y sólo si, se verifican las dos condiciones siguientes:

- **$X \rightarrow (A_1 A_2 A_3 \ldots A_n)$**
- **Ningún subconjunto propio de X, tal que Y, $Y \subseteq X$, verifica que $(Y \rightarrow (A_1 A_2 \ldots A_n))$**

En toda relación existe por lo menos una clave. En efecto, por reflexividad:

$(A_1 A_2 A_3 \ldots A_n) \rightarrow (A_1 A_2 A_3 \ldots A_n)$

Luego, o bien $(A_1 A_2 \ldots A_n)$ es una clave, o bien lo es uno de sus subconjuntos.

Añadiendo atributos a una clave se obtienen superclaves.

De la definición anterior se deduce que si X es una superclave de R, toma valores únicos en R. En efecto, como $(X \rightarrow A_1 A_2 \ldots A_n)$, si hubiera dos tuplas con valores repetidos de X, serían idénticas, lo que es imposible.

Si X e Y son superclaves, se verifica que $(X \rightarrow Y)$ y $(Y \rightarrow X)$.

Obsérvese que cuando decimos que se verifica $(X \rightarrow A_1 A_2 \ldots A_n)$, queremos decir que esta dependencia funcional está en **DF+** (aunque puede que no esté en **DF**).

Ejemplo 1

Supongamos que en la relación de Pedidos anterior sólo puede haber un proveedor por ciudad, por lo que **DF** es:

(ART, PROV)→CANT;
ART→PRECIO;
PROV→CIUDAD;
CIUDAD→KM;
CIUDAD→PROV;

Aplicando los axiomas de Armstrong deducimos:

(ART, PROV)→ART; (reflexibilidad)
(ART, PROV)→PRECIO; (transitividad)
(ART, PROV)→PROV; (reflexividad)
(ART, PROV)→CIUDAD; (transitividad)
(ART, PROV)→KM; (transitividad)

Por tanto, (ART, PROV) es una superclave. Como además es mínima, es decir, no se puede suprimir ninguno de sus componentes sin que deje de ser superclave, también es una clave.

Por otra parte, por aumentación de la última DF de **DF**, tenemos que (ART, CIUDAD)→(ART, PROV), por lo que también (ART, CIUDAD) es una superclave (y una clave, por ser mínima).

Ejemplo 2.

Supongamos que dividimos cada ciudad del país en distritos que se numeran secuencialmente, de tal modo que un distrito puede contener una ciudad completa,

o una parte de ella, y que dentro de una ciudad no haya calles o plazas con el mismo nombre. Consideremos el esquema de relación siguiente:

ZONAS (CIUDAD, DIRECCION, DISTRITO)

Las dependencias funcionales, **DF**, serían:

(CIUDAD, DIRECCION)→DISTRITO;
DISTRITO→CIUDAD;

De aquí, aplicando los axiomas de Armstrong, se deduce que hay dos claves:

(CIUDAD, DIRECCION) y (DISTRITO, DIRECCION).

PROPAGACION DE LAS DEPENDENCIAS FUNCIONALES AL DESCOMPONER

Supongamos ahora que tenemos un esquema de relación $R(A_1, A_2, \ldots, A_n)$ y que A, B, C, son conjuntos de atributos que forman una partición de U (es decir, son subconjuntos propios de U, disjuntos, y cubren totalmente a U). Sea **DF** el conjunto de condiciones de integridad aplicables a R, formado por condiciones expresables todas como dependencias funcionales, y supongamos que $(A \rightarrow B)$ está en **DF+**.

Como $(A \rightarrow B)$ implica que también se cumple que $(A \twoheadrightarrow B)$, sabemos que si descomponemos R en S y T mediante proyecciones, esta descomposición es reversible por yunción. Es decir, cualquier extensión de R da lugar, proyectándola, a dos extensiones de S y T que al ayuntarlas reproducen la de R.

Entonces el esquema de diseño:

B = (R),
DF tal que **DF+** contiene a $(A \rightarrow B)$,

lo hemos transformado en:

B′ = (S, T),

donde S = **P**(A, B) (R), T = **P**(A, C) (R), y

CINT′,

donde **CINT′** es el conjunto de condiciones de integridad aplicables a **B′**, que será consecuencia de las condiciones, **DF**, aplicables a **B**.

Vamos a ver cómo determinar este conjunto **CINT′**. Para ello veremos cómo se transforman las dependencias de **DF** al pasar de **B** a **B′**.

Supongamos que $(X \rightarrow Y)$ está en **DF+** y que $(XY \subseteq AB)$. Entonces, las tuplas de S cumplen la condición $(X \rightarrow Y)$ también. En efecto, sean s_1 y s_2 dos tuplas de S con iguales valores en X. Estas tuplas se obtienen al proyectar R a partir de unas tuplas t_1 y t_2 de R, respectivamente. Como en R se verifica $(X \rightarrow Y)$, los

valores de Y en t_1 y t_2 serán iguales. Evidentemente, también lo serán en s_1 y s_2. Luego también en S se cumple (X→Y).

Por tanto, si extraemos de **DF+** todas las dependencias funcionales cuyos atributos están incluidos en AB, tenemos un conjunto, **DFS**, que es aplicable a S. Análogamente, extrayendo de **DF+** las dependencias funcionales cuyos atributos están en AC, tendremos un conjunto, **DFT**, que es aplicable a T.

Tendremos por tanto:

- Primer esquema de diseño;

  (R, **DF**)

- Segundo esquema de diseño:

  (S, **DFS**), (T, **DFT**)

DFS y **DFT** se han formado extrayendo dependencias funcionales de **DF+**, pero es posible que haya en **DF+** dependencias funcionales que no figuren en **DFS** o **DFT**. Estas dependencias funcionales representan condiciones de integridad de **B** que no son expresables como dependencias funcionales en **B**′. Por tanto, las hemos perdido en el nuevo diseño, y en consecuencia, si no las tenemos en cuenta, cada vez que actualicemos el nuevo esquema **B**′ podemos obtener extensiones inválidas.

De modo más preciso: diremos que la descomposición de R en S y T preserva las dependencias funcionales si (DF+)=(DFS+) ∪ (DFT+).

No siempre se cumple esta condición.Vemos, pues, que es necesario que una descomposición por proyección cumpla las dos condiciones siguientes:

1) *Reversibilidad por yunción* (que nos asegura que cualquier extensión válida de R puede reconstruirse a partir de sus proyecciones).
2) *Preservación de dependencias funcionales* (que nos asegura que cualesquiera extensiones válidas de S y T, al ayuntarlas, nos producirán extensiones válidas de R).

Ejemplo:

Tomemos la relación ZONAS:

ZONAS (CIUDAD, DIREC, DIST), con **DF**:
(CIUDAD, DIREC)→DIST,
DIST→CIUDAD;

Proyectando sobre (DIST, CIUDAD) y (DIST, DIREC) obtenemos un segundo esquema de diseño:

DISTRITOS (DIREC, DIST);
AREAS (CIUDAD, DIST),
DIST→CIUDAD;

Hemos perdido la condición (CIUDAD, DIREC)→DIST.

Una extensión válida en el segundo esquema de diseño podría ser la siguiente:

DISTRITOS	DIREC	DIST
	c.A,1	10
	c.A,1	11

AREAS	CIUDAD	DIST
	BARNA	10
	BARNA	11
	MADR	20
	MADR	21

Al ayuntar ambas obtenemos:

DISTRITOS * AREAS	CIUDAD	DIREC	DIST
	BARNA	c.A,1	10
	BARNA	c.A,1	11

Esta relación no es válida en el primer esquema de diseño, pues no cumple la DF: (CIUDAD, DIREC)→DIST. En consecuencia, si decidimos aceptar el diseño representado por el segundo esquema, cada vez que haya una actualización (añadido, modificación o borrado de tuplas), habrá que comprobar que a cada valor de (CIUDAD, DIREC) sólo corresponde un número de distrito. Esto implica hacer la yunción de DISTRITOS y AREAS para verificar sobre ella si se cumple (CIUDAD, DIREC)→DIST. Por tanto sería preferible adoptar el primer esquema de diseño.

ANOMALIAS DE ACTUALIZACION

Supongamos que en R se verifica que (X→Y) y que X no es una superclave. Entonces puede haber tuplas de R con iguales valores en X. Es decir, los valores de X pueden repetirse. Esto nos plantea algunas dificultades o anomalías en las operaciones de mantenimiento.

1) *Anomalías de actualización o modificación.* Cada vez que se actualice una tupla modificando el valor de Y, hay que comprobar si hay tuplas con igual valor de X que la que estamos actualizando, y en caso afirmativo, modificar también en ellas el valor de Y, para que se siga cumpliendo X→Y. Esta verificación podría ser responsabilidad del sistema, y si no, tendrá que hacerlo el programa de aplicación.

2) *Anomalías de borrado.* Cada vez que se borre una tupla cuyo valor de X no esté repetido en ninguna otra, estamos perdiendo una información que hasta ese momento teníamos a nuestra disposición implícitamente, es decir, cuál es el valor de Y que corresponde a este valor de X. Esta información no se pierde si el valor de X está repetido en otras tuplas. Por tanto, la cantidad de información que se pierde al borrar una tupla no es siempre igual.

3) *Anomalías de inserción.* Cada vez que se inserta una nueva tupla, hay que comprobar que su valor de Y es igual al que haya en las tuplas que tengan igual valor en X que la que estamos insertando.

El hecho de que en un esquema de diseño se presenten estas anomalías no lo invalida, en principio. Lo único que ocurre es que hay que hacer comprobaciones adicionales, que pueden ser costosas en tiempo y accesos, al hacer actualizaciones. Además, si no se hacen correctamente, se pone en riesgo la integridad de los datos.

Si suponemos que X→Y y que X es una superclave, no se presentan estas anomalías. Como X no puede tomar valores repetidos, no hay que comprobar si hay otras tuplas con el mismo valor de X que la que estamos insertando o actualizando en cada momento (no hay por tanto anomalías de actualización o inserción). En cuanto a los borrados de tuplas, todos suponen la misma pérdida de información, pues al no estar repetidos los valores de X, siempre se pierde la asociación del valor de Y correspondiente al X de la tupla borrada (no hay anomalías de borrado).

En resumen, nos interesará, en principio, descomponer una relación proyectando sobre XY si se cumple que:

- X→Y, y por tanto la descomposición es reversible por yunción.
- X no es una superclave.
- La descomposición preserva las dependencias.

FORMA NORMAL DE BOYCE-CODD

Vamos a designar con el nombre de dependencias funcionales triviales a aquellas que se pueden deducir sin más que aplicar los axiomas de Armstrong al conjunto, U, de atributos. O, dicho de otra forma, son aquellas que se cumplen en todas las extensiones posibles de R (sin ninguna restricción). Así, por ejemplo, serán dependencias triviales:

(ABC)→A; (ABC)→(BC); etc.

Definición de forma normal de Boyce-Codd: Una relación R está en forma normal de Boyce-Codd si para cualquier dependencia funcional no trivial, tal que X→Y, que esté en DF+, se verifica que X es una superclave. La representaremos abreviadamente por FNBC.

De acuerdo con esta definición, las relaciones que están en FNBC no presentan las anomalías de actualización descritas en el apartado anterior.

Toda relación R que no esté en FNBC, puede transformarse mediante proyecciones en otras que sí lo estén, mediante el algoritmo siguiente:

Algoritmo de descomposición

1) Tomemos una DF, tal como X→Y, que pertenezca a **DF+** y en la que X no sea una superclave y X e Y sean disjuntos. Esto siempre es posible,

pues como R no está en FNBC, existe al menos una DF, tal que X→Z, donde el antecedente, X, no es una superclave. Es posible que X y Z sean disjuntos o no. Si no lo son, aplicando los axiomas de Armstrong podemos inferir que (X→(Z−X)), y aquí antecedente y consecuente son disjuntos. Por tanto, si R no es FNBC, siempre hay al menos una DF, X→Y, donde X no es superclave y X e Y son disjuntos.

2) Obtengamos las proyecciones de R sobre (XY) y sobre (U−Y) (donde U es el conjunto de los atributos de R). Sean S y T estas proyecciones, respectivamente. Extraemos de **DF+** las dependencias, **DFS** y **DFT**, aplicables a S y T respectivamente.

   Como (X→Y) está en **DF+** y sus atributos son los de S, también es aplicable a S. Por otra parte, por aumentación: (X→XY), luego X es una superclave de S. Es decir, (X→Y) no viola en S la condición para FNBC.

3) Si S o T no están aún en FNBC, podemos descomponerlas a su vez. Si algunas de las relaciones resultantes tampoco están en FNBC, las descomponemos a su vez, etc.

   Como en cada descomposición hemos hecho superclave al antecedente de una DF, llegará a un momento en que, o bien ya no habrá dependencias cuyos antecedentes no sean superclaves (es decir, todas las relaciones son FNBC), o bien habremos llegado a alguna relación binaria indescomponible que no esté en FNBC. Pero esto es imposible, pues si una relación binaria V(A_1, A_2) no es FNBC, quiere decir que hay una DF no trivial, $A_1 \rightarrow A_2$, donde el antecedente no es superclave, lo que es imposible porque de ($A_1 \rightarrow A_2$) se infiere ($A_1 \rightarrow A_1 A_2$) y, por tanto, A_1 es superclave.

En conclusión, cualquier relación que no esté en FNBC puede descomponerse en otras que sí lo estén. Además, por el procedimiento seguido, la descomposición es reversible por yunción. Lo que no puede asegurarse en general es que se hayan preservado en este proceso las dependencias funcionales.

Ejemplo 1:

Tomemos la relacion ZONAS ya mencionada anteriormente.

B:

ZONAS (CIUDAD, DIREC, DIST);

DF:

(CIUDAD, DIREC)→DIST,
DIST→CIUDAD;

Claves:

(CIUDAD, DIREC),
(DIST, DIREC);

Como en (DIST→CIUDAD) el antecedente no es una superclave, la relación ZONAS no está en FNBC.

Para normalizarla, la proyectamos sobre (DIST, CIUDAD) y sobre (DIST, DIREC), obteniendo las relaciones DISTRITOS y AREAS:

DISTRITOS (DIREC, DIST)
Clave: (DIREC, DIST)

AREAS (CIUDAD, DIST)
DIST→CIUDAD,
Clave: (DIST)

Tanto DISTRITOS como AREAS están en FNBC, pero, como ya vimos anteriormente, esta descomposición no preserva las dependencias. En este ejemplo sería preferible aceptar el diseño no normalizado, aunque presente anomalías de actualización.

Ejemplo 2:

B:

PEDIDOS (ART, PROV, NOMPRO, CANT)

ART:	Código de Artículo
PROV:	Código de Proveedor
NOMPRO:	Nombre de Proveedor
CANT:	Cantidad pedida

DF:

PROV→NOMPRO
ART, PROV→CANT
NOMPRO→PROV

Claves:

(ART, PROV)
(ART, NOMPRO)

Esta relación no está en FNBC, puesto que los antecedentes de las DFS (PROV→NOMPRO) y (NOMPRO→PROV) no son superclaves. Para normalizarla vamos a proyectar sobre (PRO, NOMPRO) y (ART, PROV, CANT):

PROVS:

PROVS (PROV, NOMPRO);
DF:
PROV→NOMPRO,
NOMPRO→PROV;

Claves:

(PROV),
(NOMPRO);

PEDS:

PEDS (ART, PROV, CANT);

DF:

(ART, PROV)→CANT;

Claves:

(ART, PROV);

Estas dos relaciones están en FNBC. Además, la descomposición realizada preserva las DFS. En este caso, en principio, sería aconsejable adoptar el diseño normalizado (a menos que predominen los programas de consultas de la relación PEDIDOS sobre las actualizaciones). Una importante ventaja del diseño normalizado obtenido es que puede contener más información que el primero. Por ejemplo, puede existir en PROVS una fila de un Proveedor sin que haya pedidos para él, lo que es imposible en el diseño de partida, en el que sólo se tiene información de los Proveedores que tienen Pedidos.

Ejemplo 3:

Sea el esquema:

B:

PR (P, CIUDAD, DISTN)

P: Código de Proveedor

CIUDAD: Nombre de la ciudad donde reside el Proveedor

DISTN: Distancia a que se encuentra la ciudad

DF:

P→CIUDAD

CIUDAD→DISTN

Claves:

(P)

Esta relación no está en FNBC, porque el antecedente de CIUDAD→DISTN no es una superclave. Obtenemos un segundo esquema de diseño proyectando:

PC (P, CIUDAD);

DF:

P→CIUDAD;

Claves:

(P);

CD (CIUDAD, DISTN);

DF:

CIUDAD→DISTN;

Claves:

(CIUDAD);

Ambas relaciones son FNBC, y se han preservado las DFS.

Ejemplo 4:

Vamos a añadir a la relación anterior el atributo «coste de transporte», que será dependiente de la distancia:

B:

PR (P, C, D, T)
P: Código de Proveedor.
C: Nombre de la ciudad donde reside el Proveedor
D: Distancia a que se encuentra la ciudad
T: Coste de transporte

DF:

$P \rightarrow C$
$C \rightarrow D$
$D \rightarrow T$

Claves:

(P)

No está en FNBC. Vamos a descomponer proyectando sobre (CD):

PR1 (C, D);
DF:
$C \rightarrow D$;

Claves:
(C); (Está en FNBC);

PR2 (P, C, T);
DF:
$P \rightarrow C$,
$C \rightarrow T$;

Claves:
(P); (No es FNBC);

Como PR2 no es FNBC, vamos a descomponerla a su vez en dos:

PR3 (C, T);
DF: ($C \rightarrow T$);
Claves: (C);
Es FNBC;

PR4 (P, C);
DF: $P \rightarrow C$;
Claves: (P);
Es FNBC;

Hemos llegado, pues, a un esquema de diseño con tres esquemas de relación, PR1, PR3 y PR4, donde todas las relaciones están en FNBC. Pero en el proceso de descomposición no se han preservado las DFS, pues hemos perdido la DF(D→T). Por ejemplo, una extensión válida en el segundo esquema de diseño podría ser:

PR1	C	D
	C1	D1
	C2	D2

PR3	C	T
	C1	T1
	C2	T2

PR4	P	C
	P1	C1
	P2	C2

Al ayuntar estas relaciones obtendríamos:

	P	C	D	T
	P1	C1	D1	T1
	P2	C2	D1	T2

Esta relación no es válida en el primer esquema de diseño porque no cumple la condición (D→T).

El proceso de normalización de una relación dada según el algoritmo de descomposición puede no ser único. Veámoslo en un ejemplo:

Ejemplo 5:

Tomemos de nuevo el esquema del ejemplo anterior. La primera descomposición puede hacerse proyectando sobre (DT) en vez de (CD) como antes. Obtendríamos:

PR5 (D, T);
DF: D→T;
Claves: (D);
Es FNBC;

PR6 (P, C, D);
DF: P→C, C→D;
Claves: (P);
No es FNBC;

Descompongamos ahora PR6:

PR7 (C, D);
DF: C→D;
Claves: (C);
Es FNBC;

PR8 (P, C);
DF: P→C;
Claves: (P);
Es FNBC;

Hemos obtenido un tercer esquema de diseño, con las relaciones PR5, PR7 y PR8, todas ellas normalizadas, y hemos preservado las DFS. Además es un esquema intuitivamente más lógico.

4

Muchos de los problemas relacionados con la teoría de normalización tienen la propiedad de ser «NP-completos», lo que significa que, con una gran probabilidad, son inherentemente exponenciales. Así se ha demostrado para el problema de dictaminar si una relación está o no en FNBC, y también para el problema de hallar una clave de tamaño mínimo para una relación dada.

TERCERA FORMA NORMAL

Hemos visto que una relación está en FNBC si tiene la propiedad de que para todas las DFs no triviales incluidas en **DF+**, sus antecedentes son superclaves. Podemos relajar algo esta condición y llegar así a otras formas normales. Veamos la tercera forma normal (FN3).

Supongamos que admitimos que en algunas dependencias los antecedentes no sean superclaves, siempre que los consecuentes sean atributos incluidos en alguna clave. Entonces la relación está en tercera forma normal.

Para precisar el enunciado anterior vamos a definir como atributos principales («prime attributes») a aquellos que participan en alguna clave.

Definición

Diremos que una relación está en tercera forma normal (FN3) si para toda DF no trivial de DF+ se verifica alguna de las condiciones siguientes:

- **O bien el antecedente es una superclave.**
- **O bien el antecedente no es una superclave y el consecuente es un conjunto de atributos principales.**

Es evidente, de acuerdo con esta definición, que toda relación que esté en FNBC también está en FN3. El recíproco no es siempre cierto. Veamos algunos ejemplos.

Ejemplo 1:

Tomemos la relación ZONAS:

B : ZONAS (CIUDAD, DIREC, DIST)
DF: (CIUDAD, DIREC)→DIST; DIST→CIUDAD.

Claves: (CIUDAD, DIREC); (DIST, DIREC)

El antecedente de (DIST→CIUDAD) no es una superclave, luego ZONAS no está en FNBC. Pero el consecuente es un atributo principal (forma parte de la clave (CIUDAD, DIREC)), luego sí está en FN3.

Ejemplo 2:

Tomemos la relación PEDIDOS, ya mencionada anteriormente:

B:

PEDIDOS (ART, PROV, NOMPRO, CANT)

ART:	Código de Artículo
PROV:	Código de Proveedor
NOMPRO:	Nombre de Proveedor
CANT:	Cantidad pedida

DF:

PROV→NOMPRO
ART, PROV→CANT
NOMPRO→PROV

Claves: (ART, PROV); (ART, NOMPRO)

Esta relación no está en FNBC, puesto que los antecedentes de las DFs (PROV→ NOMPRO) y (NOMPRO→PROV) no son superclaves, pero sí está en FN3 porque los consecuentes son atributos principales.

Ejemplo 3:

Tomemos la relación PR, ya mencionada anteriormente:

B:

PR (P, C, D, T)

P: Código de Proveedor
C: Nombre de la ciudad donde reside
D: Distancia a que se encuentra ésta
T: Coste de transporte

DF:

(P→C); (C→D); (D→T)

Claves: (P)

No está en FNBC ni en FN3.

Toda relación que no esté en FN3 puede descomponerse en otras que sí lo estén mediante proyecciones reversibles por yunción, aplicando el mismo algoritmo de descomposición expuesto anteriormente para descomponer en FNBC. Recordémoslo:

Algoritmo de descomposición

1) Tomemos una DF, X→Y, donde X e Y sean disjuntos y X→Y viole las condiciones de FN3, es decir, X no sea superclave e Y contenga algún atributo no principal.

   Esto siempre es posible.

2) Obtengamos (S = **P**(XY) (R)) y (T = **P**(U − Y) (R)). Como ya vimos, X→Y no viola ya las condiciones de FN3 en S, porque X es una superclave de S.

3) Si S o T no son FN3, se vuelven a descomponer, etc.

4) En cada una de estas descomposiciones se ha eliminado una DF que viola las condiciones de FN3. Como hay un número finito de DFs, llegará un momento en que no habrá ninguna DF que viole las condiciones o tendremos relaciones binarias. Pero éstas son FN3 (puesto que, como ya vimos, son FNBC).

Este algoritmo, como ya vimos al hablar de FNBC, produce descomposiciones reversibles por yunción, pero puede no preservar las DFs.

Ejemplo 4:

Descomposición en FN3. Tomemos la relación de Pedidos siguiente:

PE (A, P, NP, CAN, CIU)
A: Código de Artículo
P: Código de Proveedor
NP: Nombre de Proveedor
CAN: Cantidad pedida
CIU: Ciudad del Proveedor

DF:

A, P→CAN
P →NP
NP →P
P →CIU

Claves: (A, P); (A, NP)

La DF (P→CIU) viola las condiciones de FN3. Proyectemos sobre (P, CIU), por tanto, para obtener PE1 y PE2:

PE1 (P, CIU);
DF: (P→CIU);
Clave: (P)
Es FN3 (y FNBC)

PE2 (A, P, NP, CAN);
DF: (A, P→CAN), (P→NP), (NP→P);
Claves: (A, P), (A, NP);
Es FN3 (pero no FNBC)

Hemos llegado a un diseño (PE1, PE2), que es reversible por yunción y preserva las DFs.

Como PE2 no es FNBC, convendría descomponerla a su vez, y obtendríamos el diseño:

PE1 (P, CIU),
PROVS (P, NP),
PE3 (A, P, CAN);

donde todas las relaciones son FNBC y además se preservan las DFs.

Sin embargo, este diseño, obtenido aplicando el algoritmo de descomposición directamente sobre las DFs dadas, es mejorable si lo aplicamos sobre DFs deducidas de las dadas. En efecto, de (P→NP) y (P→CIU) se infiere que (P→NP, CIU). Esta DF viola las condiciones de FN3 y de FNBC, pues el antecedente no es una superclave y el consecuente, (NP, CIU), tiene un atributo no principal.

Vamos, pues, a descomponer proyectando sobre esta DF, obteniendo PE4 y PE5:

PE4 (P, NP, CIU);
DF: (P→NP), (NP→P), (P→CIU);
Claves: (P), (NP);
Es FNBC.

PE5 (A, P, CAN);
DF: (A, P→CAN);
Clave: (A, P);
Es FNBC.

Este diseño es reversible por yunción, preserva las DFs y sólo tiene dos relaciones. Es por tanto preferible al anterior (formado por PE1, PROVS y PE3). Además, es intuitivamente más lógico.

DEPENDENCIAS PARCIALES Y TRANSITIVAS

Diremos que un atributo, A, depende transitivamente de una clave X cualquiera si A es no principal y existe una dependencia (Y→A), donde Y no es una superclave ni parte de una clave.

Diremos que un atributo A depende parcialmente de una clave X, si A es no principal y existe una dependencia (Y→A), donde Y es parte de la clave X.

De acuerdo con estas definiciones, si una relación está en FN3, no tiene dependencias parciales ni transitivas, pues si hubiera alguna se violarían las condiciones que debe cumplir una relación FN3.

Recíprocamente, si no hay dependencias parciales ni transitivas, la relación está en FN3. En efecto, si no lo estuviera, habría alguna DF, tal que Y→A, donde Y no fuera superclave y A fuera no principal, es decir, habría alguna dependencia parcial o transitiva.

En resumen, una relación está en FN3 si, y sólo si, no tiene dependencias parciales ni transitivas.

DESCOMPOSICION PRESERVANDO LAS DEPENDENCIAS FUNCIONALES

Sea una relación, R, y sea **DF** el conjunto de sus dependencias funcionales. Supongamos que **DF** es un cubrimiento mínimo de **DF**+, es decir, que cumple las condiciones siguientes:

1) Todas las dependencias tienen un solo atributo en su consecuente (es decir, detrás de la flecha). Si así no fuera se descompone. Por ejemplo, X→AB se descompondría en X→A y X→B.

2) No hay en **DF** dependencias redundantes. O lo que es lo mismo, ninguna DF de **DF** puede inferirse de las restantes, o lo que es igual, si se suprime de **DF** una DF cualquiera, ya no podrían inferirse de las que queden todas las DFs de **DF+**.
3) No hay atributos redundantes en ningún antecedente de **DF**. Es decir, que no se puede suprimir ningún atributo en la parte izquierda de las dependencias de **DF** sin que esto afecte a **DF+**.

Vamos a descomponer R mediante el siguiente procedimiento.

Algoritmo de descomposición que perserva dependencias

- Entrada: R.
- Salida: n relaciones obtenidas a partir de R.
- Proceso:
  1) Si hay atributos en R que no figuran en los antecedentes o consecuentes de **DF**, proyectar R sobre ellos. Poner la relación resultante en la salida.
  2) Si en alguna DF de **DF** figuran todos los atributos de R (bien en el antecedente, bien en el consecuente), poner R en la salida.
  3) Si hay en **DF** varias DFs con el mismo antecedente, tal como X→A, X→B, ..., etc., proyectar R sobre (X, A, B, ...). Poner la relación resultante en la salida.
  4) Tomar una DF de **DF** cuyo antecedente no esté repetido en ninguna otra, tal como Y→D, y proyectar R sobre (Y, D). Poner la relación resultante en la salida.
  5) Repetir los pasos 3 y 4 anteriores hasta que se hayan tratado todas las DFs de **DF**. Cuando se haya terminado con la última, pasar al punto siguiente.
  6) Proyectar R sobre una cualquiera de sus claves. Poner la relación resultante en la salida. Fin del algoritmo.

Es evidente que este algoritmo preserva las DFs. Puede demostrarse además que es reversible por yunción y que todas las relaciones obtenidas son FN3. Por tanto, puede ser preferible al algoritmo de descomposición descrito anteriormente. El problema con este algoritmo es que suele producir relaciones redundantes.

Ejemplo 1:

Sea el esquema:

R(A, B, C);
DF: B→C;
Clave: (A, B).

Aplicando el algoritmo tendríamos:

R1(A) (puesto que A no participa en las DFs dadas)
R2(B, C)
R3(A, B) (por ser (AB) una clave)

Evidentemente, R1(A) es redundante con R3(A, B), puesto que ésta contiene los atributos de aquélla. Tendremos, en definitiva, el diseño siguiente:

R2(B, C),
(B→C),
FNBC.

R3(A, B),
Clave: (A, B),
FNBC.

Ejemplo 2:

Sea el esquema:

R(A, B, C, D, E, F, G);
DF: B, C→D,
E →F,
E →G;
Clave: (A, B, C, E).

Aplicando el algoritmo se puede descomponer en:

R4 (A),
FNBC.

R5 (B, C, D),
BC→D,
Clave: (BC),
FNBC.

R6 (E, F, G),
(E→F), (E→G),
Clave: E,
FNBC.

R7(A, B, C, E),
FNBC.

Evidentemente, R4 es redundante con R7.

Ejemplo 3:

CURSO (AS, P, AU, H, AL, C)

AS: Asignatura
P: Profesor
AU: Aula
H: Hora
AL: Alumno
C: Calificación

Conjunto de dependencias, **DF**:

- Cada asignatura tiene un solo profesor, (AS→P).
- En un momento dado, un aula está ocupada por una sola asignatura, (H, AU→AS).
- En un momento dado, un profesor sólo puede estar explicando en un aula, (H, P→AU).
- Cada estudiante tiene una sola calificación por asignatura, (AS, AL→C).
- En un momento dado, un estudiante sólo puede estar en un aula, (H, AL→AU).

Sólo hay una clave: (H, AL). Además **DF** es un cubrimiento mínimo de **DF**+.

Vamos a normalizar la relación CURSO, según el algoritmo de descomposición que nos permite la obtención de relaciones en FNBC. El proceso es el siguiente (se señala en cada paso con un * la DF elegida para proyectar):

Relación de partida:

CURSO (AS, P, AU, H, AL, C)
AS →P
H AU→AS
H P→AU
* AS AL→C
H AL→AU
Clave: (H, AL)

Proyectando sucesivamente se obtiene:

CUR1 (AS, AL, C)
AS AL→C
Clave: (AS, AL)
FNBC

CUR2 (AS, P, AU, H, AL)
* AS →P
H AU→AS
H P →AU
H AL→AU
Clave: (H, AL)

CUR3 (AS, P)
AS →P
Clave: (AS)
FNBC

CUR4 (AS, AU, H, AL)
H AU→AS
* H AS →AU
H AL→AU
Clave: (H, AL)

CUR5 (AS, AU, H)
AS H →AU
AU H →AS
Claves: (AS, H) y (AU, H)
FNBC

CUR6 (AS, H, AL)
H AL →AS
Clave: (H, AL)
FNBC.

En resumen, hemos llegado así al diseño:

CUR1 (AS, AL, C): Calificaciones por alumno y asignatura.

CUR3 (AS, P): Profesores de las asignaturas.

CUR5 (AS, AU, H): Horarios y aulas por asignatura.

CUR6 (AS, H, AL): Alumnos por cada asignatura y hora.

En el proceso de descomposición hemos perdido la DF (H, P→AU), al pasar de CUR2 a CUR4.

Apliquemos ahora a CURSO el algoritmo de descomposición que preserva dependencias. Obtenemos el siguiente esquema:

CUR7 (AS, P)
AS →P
Clave: (AS)
FNBC

CUR8 (AS, AU, H)
AS H →AU
AU H →AS
Claves: (AS, H) y (AU, H)
FNBC

CUR9 (P, AU, H)
H P →AU
H AU→P
Claves: (H, P) y (H, AU)
FNBC

CUR10 (AS, AL, C)
AS AL →C
Clave: (AS, AL)
FNBC

CUR11 (AU, H, AL)
H AL →AU
Clave: (H, AL)
FNBC

CUR12 (H, AL)
Clave: (H, AL)
FNBC.

Evidentemente, CUR12 está contenida en CUR11. Además, CUR7 es igual que CUR3, CUR8 que CUR5 y CUR10 que CUR1, por lo que queda finalmente el diseño:

CUR1 (AS, AL, C): Calificaciones por alumno y asignatura.

CUR3 (AS, P): Profesores de las asignaturas.

CUR5 (AS, AU, H): Horarios y aulas por asignatura.

CUR9 (P, AU, H): Horarios y aulas por profesor.

CUR11 (AU, H, AL): Alumnos por aula y hora.

Este diseño contiene más redundancia que el anterior, pero preserva las dependencias y aquél no.

SEGUNDA FORMA NORMAL

Se define relajando algo las condiciones de FN3. Vamos a describirla a continuación por su interés histórico.

Una relación está en segunda forma normal (FN2) si para toda DF de DF+ no trivial se verifica una de las condiciones siguientes:

- **O bien el antecedente es una superclave.**
- **O bien el antecedente no es una superclave y el consecuente es un conjunto de atributos principales.**
- **O bien, si el antecedente no es una superclave ni el consecuente un conjunto de atributos principales, el antecedente no es parte de ninguna clave.**

Es evidente, por la definición anterior, que toda relación que esté en FN3 también está en FN2. El recíproco no es cierto.

Si una relación no tiene dependencias parciales, está en FN2, pues si así no fuera, habría una dependencia, Y→A, donde Y no sería una superclave, A sería no principal e Y sería parte de alguna clave (para violar las tres condiciones que definen una FN2). Es decir, habría una dependencia Y→A, que sería una dependencia parcial, lo que contradice la hipótesis.

Recíprocamente, si una relación está en FN2, no tiene dependencias parciales, pues si hubiera alguna contradiría las condiciones anteriores.

En resumen, para que una relación esté en FN2 es condición necesaria y suficiente que no tenga dependencias parciales (aunque sí puede tener dependencias transitivas).

Sólo para completar el panorama mencionaremos la existencia de lo que se conoce como primera forma normal. Una relación está en esta forma normal cuando sus atributos son indescomponibles (es decir, no hay subcampos dentro de campos). Nosotros, al definir las relaciones, hemos considerado

dominios formados por conjuntos de valores indescomponibles, es decir, que no son a su vez conjuntos. Por ello, desde el principio hemos implícitamente supuesto que todas las relaciones a que nos referíamos eran en primera forma normal.

Toda relación que no esté en FN2 puede descomponerse en otras que sí lo estén, mediante proyecciones.

Una relación que esté en FN2 puede presentar anomalías de actualización suprimibles al descomponerla en relaciones FN3. También al descomponer una FN3 en otras FNBC se pueden eliminar anomalías. Por tanto, en principio, interesa descomponer en FNBC si se preservan las dependencias en la descomposición. Si no, se puede al menos descomponer en relaciones FN3, preservando las dependencias.

Ejemplo 1:

Tomemos la relación PR, ya comentada anteriormente:

B: PR (P, C, D, T)

P: Código de Proveedor
C: Nombre de la ciudad donde reside
D: Distancia a que se encuentra ésta
T: Coste de transporte

DF: (P→C); (C→D); (D→T)

Claves: (P)

No está en FNBC ni en FN3, pero sí está en FN2 (los antecedentes de C→D y D→T no forman parte de ninguna clave).

Ejemplo 2:

Tomemos ahora la relación de Pedidos, PE, anteriormente comentada:

PE (A, P, NP, CAN, CIU)

A: Código de Artículo
P: Código de Proveedor
NP: Nombre de Proveedor
CAN: Cantidad pedida
CIU: Ciudad del Proveedor

DF: (A, P)→CAN; P→NP; NP→P; P→CIU

Claves: (A, P); (A, NP)

No es FNBC ni FN3. Tampoco es FN2 porque en (P→CIU) el antecedente es parte de una clave.

DEPENDENCIAS PLURALES

Hemos visto que, dada una relacioın R y un conjunto de DFs definidas sobre ella, se puede descomponer en relaciones FNBC o FN3 con reversibilidad por yunción.

Vamos ahora a plantearnos el mismo problema, pero con condiciones de integridad más generales que las DFs. Vamos a considerar el caso en que nuestro conjunto de condiciones de integridad esté formado por DFs y DPs (o sea, dependencias funcionales y plurales). Representaremos con **DEP** al conjunto dado de DFs y DPs que se verifican en la relación.

Recordemos que $A \twoheadrightarrow B$ significa que si hay en R dos tuplas con el mismo valor en A, también existen en R las que resulten de intercambiar entre ellas los valores de B.

SISTEMA DE INFERENCIA PARA DEPENDENCIAS PLURALES

Análogamente a los axiomas de Armstrong para DFs, existen unos axiomas aplicables a las DPs y DFs que permiten inferir unas de otras. Estos axiomas forman también un sistema de inferencias sano y completo (es decir, todas las dependencias inferibles lógicamente a partir de **DEP** pueden deducirse aplicándoles los axiomas, y recíprocamente). Sea una relación R, con un conjunto U de atributos, con un conjunto dado, **DEP**, de dependencias funcionales y plurales, y sean X, Y, Z, V, W, subconjuntos de U. Los axiomas son los siguientes:

1) *Reflexividad de DFs:* Si $(Y \subseteq X)$, entonces $X \rightarrow Y$.
2) *Aumentación de DFs:* Si $(X \rightarrow Y)$, entonces $(XZ \rightarrow YZ)$.
3) *Transitividad de DPs:* Si $(X \rightarrow Y)$ y $(Y \rightarrow Z)$, entonces $(X \rightarrow Z)$.
4) *Complementación de DPs:* Si $(X \twoheadrightarrow Y)$, entonces $X \twoheadrightarrow (U - X - Y)$.
5) *Aumentación de DPs:* Si $(X \twoheadrightarrow Y)$ y $(V \subseteq W)$, entonces $(WX \twoheadrightarrow VY)$.
6) *Transitividad de DPs:* Si $(X \twoheadrightarrow Y)$ y $(Y \twoheadrightarrow Z)$, entonces $(X \twoheadrightarrow Z - Y)$.
7) *Paso de DF a DP:* Si $(X \rightarrow Y)$, entonces $(X \twoheadrightarrow Y)$.
8) *Paso de DP a DF:* Si $(X \twoheadrightarrow Y)$ y Z es una parte de Y (es decir, $Z \subseteq Y$) y $(W \rightarrow Z)$, siendo Y y W disjuntos, entonces se cumple que $(X \rightarrow Z)$.

De estos axiomas pueden deducirse otras propiedades interesantes, como las siguientes:

1) Reflexividad para DPs: Si $(Y \subseteq X)$, sabemos que $X \rightarrow Y$, luego también se cumple que $X \twoheadrightarrow Y$.
2) Si $(X \twoheadrightarrow Y)$ y $(X \twoheadrightarrow Z)$, entonces $(X \twoheadrightarrow Y - Z)$. Puede comprobarse mediante los pasos siguientes:

   $X \twoheadrightarrow Y$; $X \twoheadrightarrow Z$; $XZ \twoheadrightarrow YZ$; $X \twoheadrightarrow XZ$; $X \twoheadrightarrow YZ - XZ$; $YZ - XZ = Y - X - Z$; $X \twoheadrightarrow Y - X - Z$; $X \cup (X \cap (Y - Z)) = X$; $(Y - X - Z) \cup (X \cap (Y - Z)) = Y - Z$; aumentando ambos lados de la última DP con $(X \cap (Y - Z))$, queda $X \twoheadrightarrow Y - Z$.

3) Regla de descomposición: Si $(X \twoheadrightarrow Y)$ y $(X \twoheadrightarrow Z)$, entonces se cumple que $(X \twoheadrightarrow Y - Z)$, $(X \twoheadrightarrow Z - Y)$ y $(X \twoheadrightarrow Y \cap Z)$. Se deduce fácilmente de la propiedad anterior. El nombre de esta regla se ve mejor si se pone en la forma:

   Si $(X \twoheadrightarrow YZ)$ y $(X \twoheadrightarrow ZV)$, siendo Y, Z y V, disjuntos, entonces $(X \twoheadrightarrow Y)$, $(X \twoheadrightarrow Z)$ y $(X \twoheadrightarrow V)$.

4) Si $(X \twoheadrightarrow Y)$, entonces $(X \twoheadrightarrow Y - X)$.
5) Complementación del consecuente: Si $(X \twoheadrightarrow Y)$, entonces $(X \twoheadrightarrow U - Y)$.
6) Regla de unión: Si $(X \twoheadrightarrow Y)$ y $(X \twoheadrightarrow Z)$, entonces $(X \twoheadrightarrow YZ)$.

Llamaremos DP triviales a las que se cumplen en cualquier extensión de R que pueda construirse, sin ninguna restricción. Dicho de otra forma, son triviales todas las DFs y DPs que se puedan inferir aplicando los axiomas a U. Para toda DP, $(X \twoheadrightarrow Y)$, trivial, se cumple que, o bien $(Y \subseteq X)$, o bien $(XY = U)$.

Por ejemplo, son triviales:

$U \twoheadrightarrow X$; $XY \twoheadrightarrow X$; $XY \twoheadrightarrow Y$; $X \twoheadrightarrow X \cap Y$; etc.

Es más interesante el caso en que $(XY = U)$. Comprobemos que, en este caso, $X \twoheadrightarrow Y$ es trivial:

$X \twoheadrightarrow X$; $X \twoheadrightarrow U - X$; $X \twoheadrightarrow YX - X$; $X \twoheadrightarrow Y - X$; $X \twoheadrightarrow X \cap Y$; $X \twoheadrightarrow (Y - X) \cup (Y \cap X)$, de donde, finalmente, $X \twoheadrightarrow Y$.

Análogamente a como hacemos con las DFs, podemos definir la clausura de **DEP**, y la representaremos **DEP**+, como el conjunto de todas las dependencias inferibles de **DEP** aplicando los axiomas. Dos conjuntos de dependencias, **DEP** y **DEP**′ serán equivalentes si **DEP**+ = **DEP**′+.

TRANSFORMACION DE LAS DEPENDENCIAS PLURALES AL DESCOMPONER

Análogamente a como hacíamos con las DFs, vamos a ver cómo aplicar las dependencias plurales de una relación a las que resulten de descomponerla en otras mediante proyecciones.

Sea una relación R, con un conjunto **DEP** de DFs y DPs. Supongamos que S es una relación obtenida proyectando R. Sea U el conjunto de los atributos de R y V el de los atributos de S. Entonces:

- Para toda DF, $X \rightarrow Y$, de **DEP**+, tal que $(X \subseteq V)$, la DF$(X \rightarrow Y \cap V)$ se verifica en S.
- Para toda DP, $X \twoheadrightarrow Y$, de **DEP**+, tal que $(X \subseteq V)$, la DP$(X \twoheadrightarrow Y \cap V)$ se verifica en S.

Veamos algunos ejemplos.

Ejemplo 1:

Tomemos la relación CURSO ya estudiada anteriormente. Además de las DFs ya comentadas, parece lógico, de acuerdo con el significado de los atributos, que a cada asignatura correspondan varias horas y aulas, independientemente de los profesores y alumnos que asistan a la clase. Esta condición equivale a decir que la asignatura pluridefine a la hora y aula, o sea, (AS→»H, AU). En definitiva, el esquema de relación de CURSO será ahora:

CURSO (AS, P, AU, H, AL, C)

(Atributos: ASignatura, Profesor, AUla, Hora, ALumno, Calificación.)

DEP:

AS →P
H AU→AS
H P →AU
AS AL →C
H AL →AU
AS →»H AU

Clave: (H, AL)

Obsérvese que de estas dependencias se infiere:

AS→»H AU;
AS→»P AL C; (complementación)
AS→P; AS→»P;
AS→»AL C;

es decir, a una asignatura corresponden varios valores de la pareja (AL, C), independientemente de los profesores y aulas.

Con estas condiciones podríamos tener un cuadro de valores como el siguiente:

	Aula 1				*Aula* 2			
Hora	*Asig.*	*Prof.*	*Alum.*	*Calif.*	*Asig.*	*Prof.*	*Alum.*	*Calif.*
9-10	Latín	P1	A11	E	Matem.	P2	A13	A
			A12	A			A14	S
10-11	Gramát.	P4	A11	A	Química	P3	A13	A
			A12	S			A14	E
11-12	Latín	P1	A11	E	Matem.	P2	A13	A
			A12	A			A14	S
12-13	Histor.	P1	A11	A	Física	P3	A13	E
			A12	E			A14	E

La extensión de la relación CURSO correspondiente a estos valores sería:

CURSO	AS	H	AU	AL	C	P
	Latín	9-10	Au1	A11	E	P1
	Latín	9-10	Au1	A12	A	P1
	Latín	11-12	Au1	A11	E	P1
	Latín	11-12	Au1	A12	A	P1
	Matem.	9-10	Au2	A13	A	P2
	Matem.	9-10	Au2	A14	S	P2
	Matem.	11-12	Au2	A13	A	P2
	Matem.	11-12	Au2	A14	S	P2
	Gramát.	10-11	Au1	A11	A	P4
	Gramát.	10-11	Au1	A12	S	P4
	Física	12-13	Au2	A13	E	P3
	Física	12-13	Au2	A14	E	P3
	Química	10-11	Au2	A13	A	P3
	Química	10-11	Au2	A14	E	P3
	Histor.	12-13	Au1	A11	A	P1
	Histor.	12-13	Au1	A12	E	P1

Se ve que efectivamente (AS→→H AU), pues si tomamos dos filas con el mismo AS:

Latín 9-10 Au1 A11 E P1,
Latín 11-12 Au1 A12 A P1,

e intercambiamos entre ellas los valores de H y AU, se obtienen otras dos filas de la relación:

Latín 11-12 Au1 A11 E P1,
Latín 9-10 Au1 A12 A P1.

Para normalizar esta relación, vimos que se podía empezar proyectando sobre (AS, AL, C) y (AS, P, AU, H, AL). Veamos cómo se propagan las DFs y DPs a las relaciones resultantes de estas proyecciones:

CUR1 (AS, AL, C)
AS AL→C
Clave: (AS, AL);

CUR2 (AS, P, AU, H, AL)
AS →P
H AU →AS
HP →AU
H AL →AU
AS →→H AU

Ejemplo 2:

Supongamos ahora que existe otra condición de integridad adicional sobre la relación CURSO que es la siguiente: un alumno no recibe más de una clase al día de una asignatura dada. Si suponemos que el horario de clases se repite diariamente, esta condición equivale a decir que (AL, AS) toma valores únicos, es decir, es una superclave. Por tanto: (AL AS→P AU H C). Como (H AL) es

clave, de (AL AS→H) se deducen las demás (AL AS→P; AL AS→AU;AL AS→C).

Tenemos entonces que **DEP** es:

AS → P
H AU → AS
H P → AU
AS AL → C
H AL → AU
AL AS → H
AS ↠ H AU

Aplicando los axiomas de inferencia:

De AS↠H AU y AS AL→AU, y teniendo en cuenta que (H AU) y (AS AL) son disjuntos, se deduce que (AS→AU).

Análogamente, de AS↠H AU y AS AL→H se deduce AS→H.

Por tanto, AS→H AU.

Es decir, la DP dada es en realidad una DF. Su significado es que una asignatura sólo se explica una vez al día, en un aula y a una hora determinadas.

Tendremos en resumen que **DEP** es equivalente a:

AS → P
H AU → AS
H P → AU
AS AL → C
H AL → AU
AS → H
AS → AU
(Claves: (H AL), (AS AL))

ANOMALIAS DE ACTUALIZACION CON DPs

Supongamos que en R se verifica (X↠Y) y que X no es una superclave. Análogamente a como razonábamos en el caso de DFs, se presentan anomalías de actualización por poder tomar X valores repetidos (y, por tanto, poder haber información redundante).

Ejemplo:

Tomemos la relación CURSO mencionada en el Ejemplo 1 de las páginas precedentes, en la que (AS↠H AU) y AS no es una superclave.

Si queremos actualizar la calificación de Latín del alumno A11, hay que hacerlo en varias filas (en todas las que aparezca el alumno A11 con la asignatura de Latín). (Anomalías de actualización.)

Si se decide dar en el centro una clase más de una asignatura, por ejemplo Historia en el aula 1 de 13 h. a 14 h., no se puede registrar esta información en la relación

CURSO hasta que haya un profesor y, al menos, un alumno matriculado para esta clase. (Anomalía de inserción.)

Análogamente, al borrar unas filas puede perderse más información que al borrar otras. (Anomalía de borrado.)

CUARTA FORMA NORMAL

Sea una relación R, con un conjunto **DEP** de DFs y DPs.

Decimos que R está en cuarta forma normal (FN4) si para toda DP no trivial, tal que $X \twoheadrightarrow Y$, de DEP+, el antecedente X es una superclave de R.

Como $X \rightarrow Y$, implica que $X \twoheadrightarrow Y$, toda relación que está en FN4 también está en FNBC.

Toda relación R se puede descomponer en relaciones FN4 de forma reversible por yunción. El algoritmo es el mismo que se describió anteriormente para descomponer en FNBC. Lo resumimos a continuación:

Algoritmo de descomposición

1) Tomar una DP de **DEP+** cuyo antecedente no sea superclave. Sea esta DP, por ejemplo $X \twoheadrightarrow Y$, y sean X e Y disjuntos.

   Si X e Y no fueran disjuntos, tomaríamos la DP $(X \twoheadrightarrow Y - X)$ en vez de $X \twoheadrightarrow Y$, y en la que antecedente y consecuente son disjuntos.

2) Obtener $S = \mathbf{P}(X, Y)(R)$ y $T = \mathbf{P}(U - Y)(R)$. Hallar las DFs y DPs aplicables a S y T a partir de **DEP+**. La DP de partida, $X \twoheadrightarrow Y$, es aplicable a S, pero no viola ya la condición de FN4 porque es trivial.

3) Si S o T no son FN4, aplicarles el mismo proceso.

Ejemplo 1:

Tomemos la relación CURSO:

CURSO (AS, P, AU, H, AL, C)

(Atributos: ASignatura, Profesor, AUla, Hora, ALumno, Calificación.)

DEP:

AS $\rightarrow$ P
H AU $\rightarrow$ AS
H P $\rightarrow$ AU
AS AL $\rightarrow$ C
H AL $\rightarrow$ AU
AS $\twoheadrightarrow$ H AU

Clave: (H, AL)

Para normalizar proyectaremos según los pasos siguientes:

CUR13 (AS, AU, H)
H AU→AS
H AS →AS
Claves: (H AU), (H AS)
FN4

CUR14 (AS, P, AL, C)
* AS →P
AS AL→C
Clave: (AS AL)
No es FNBC ni FN4

CUR15 (AS, P)
AS→P
Clave: (AS)
FN4

CUR16 (AS, AL, C)
AS AL→C
Clave: (AS AL)
FN4

Hemos perdido en la descomposición las DFs (H P→AU) y (H AL→AU). El diseño obtenido es:

CUR13 (AS, AU, H); CUR15 (AS, P); CUR16 (AS, AL, C)

Como hemos perdido DFs, este diseño podría generar filas inválidas al ayuntar. Por ejemplo:

CUR13	AS	AU	H
	Latín	Au1	9-10
	Histor.	Au2	9-10

CUR15	AS	P
	Latín	P1
	Matem.	P2
	Histor.	P1

CUR16	AS	AL	C
	Latín	A11	E
	Latín	A12	A
	Histor.	A13	S

La yunción de estas tres relaciones da:

AS	AU	H	P	AL	C
Latín	Au1	9-10	P1	A11	E
Latín	Au1	9-10	P1	A12	A
Histor.	Au2	9-10	P1	A13	S

Vemos que en esta última relación no se cumple la DF ($H, P \rightarrow AU$) (en la relación parece que el profesor P1 estaría a la vez en el aula 1 y en la 2 a la misma hora, dando clases diferentes, una de Latín y la otra de Historia).

Ejemplo 2:

Supongamos que tenemos dos máquinas para fabricar tres tipos de productos, de modo que cada máquina puede fabricar varios de éstos, por ejemplo:

- La máquina M1 sirve para fabricar los productos P1 y P2.
- La máquina M2 sirve para fabricar los productos P2 y P3.

Cada empleado sólo sabe manejar una máquina. Por ejemplo:

- El empleado E1 sabe utilizar la máquina M1.
- El empleado E2 sabe utilizar la máquina M2.
- El empleado E3 sabe utilizar la máquina M2.

Podemos construir una relación, FABRICA, que indique qué productos puede fabricar cada empleado en cada máquina: FABRICA (E, M, P). (Atributos: Empleado, Máquina, Producto.)

La extensión de esta relación en nuestro ejemplo será:

FABRICA	E	M	P
	E1	M1	P1
	E1	M1	P2
	E2	M2	P2
	E2	M2	P3
	E3	M2	P2
	E3	M2	P3

Las condiciones especificadas pueden representarse con las dependencias: $E \rightarrow M$; $M \twoheadrightarrow P$.

Es decir, tenemos el esquema:

FABRICA (E, M, P)
$E \rightarrow M$
* $M \twoheadrightarrow P$
Clave: (E, P).

No es FNBC ni, por tanto, FN4. Normalicemos:

FAB1 (M, P)
    Clave: (M, P)
    FN4

FAB2 (E, M)
    E→M
    Clave: (E)
    FN4

En el ejemplo dado las extensiones serían:

FAB1	M	P
	M1	P1
	M1	P2
	M2	P2
	M2	P3

FAB2	E	M
	E1	M1
	E2	M2
	E3	M2

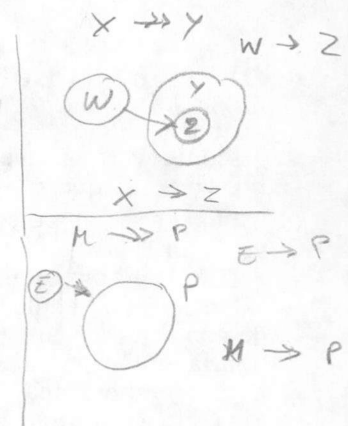
Ejemplo 3:

Supongamos ahora que en la relación anterior se cumple otra condición más: que cada empleado sólo sabe fabricar un tipo de producto. Tendríamos:

FAB (E, M, P)
    E →M
    E →P
    M→→P
    Clave: (E).

De aquí se infiere la DF (M→P) como sigue:
    M→→P; E→P; como E y P son disjuntos: M→P.

Es decir, la DP (M→→P) es en realidad una DF.



Tendremos pues:

FAB (E, M, P)
    E →M
  * M→P
    Clave: (E)
    No es FNBC

Normalizando tendríamos;

FAB3 (M, P)
    M→P
    Clave: (M)
    FNBC y FN4

FAB4 (E, M)
    E→M
    Clave: (E)
    FNBC y FN4

Ejemplo 4:

Supongamos ahora que en la relación anterior no se verifica ninguna DF, es decir, que un empleado puede usar varias máquinas y fabricar todos los tipos de productos que se puedan fabricar en esas máquinas.

Tendríamos:

FABRI (E, M, P)
M—»P
Clave: (E, M, P)
FNBC (no FN4)

Por complementación: M—»E.

Normalizando:

FABRI1 (M, P)
Clave: (M, P)
FN4

FABRI2 (E, M)
Clave: (M, E)
FN4

Aparentemente han desaparecido las DPs originales, (M—»P) y (M—»E). No es así, porque son DPs triviales en FABRI1 y FABRI2, respectivamente. Cualesquiera que sean las filas que insertemos en FABRI1 y FABRI2, la yunción de éstas producirá una extensión válida de FABRI, pues al asociar todos los valores de P y E correspondientes a uno de M, se cumple M—»P (y M—»E) en el resultado de la yunción.

DEPENDENCIAS EMBEBIDAS

En el último ejemplo anterior hemos visto que las DPs se han propagado al descomponer, aunque se han transformado en triviales. Pero, por otra parte, la DP (P—»M), que se verifica en FABRI1 (es trivial), no es válida en FABRI.

Es decir, al descomponer una relación pueden aparecer DPs que no se verifican en la relación original. Si estas DPs son triviales, como la (P—»M) de FABRI1, no importa, porque no suponen ninguna restricción, es decir, cualquier extensión las cumple. Si no son triviales, estamos en presencia de un nuevo tipo de condición de integridad que no es expresable como DF ni como DP, y que podría enunciarse diciendo que las tuplas de la relación original deben ser tales que se cumpla una determinada DP en el resultado obtenido al proyectarla. Esta situación no se puede presentar con DFs porque cualquier DF aplicable a una proyección lo es también a la relación original, y en cambio para las DPs puede no ser así.

Con más precisión, sea una relación R, con dependencias **DEPR**. Sea S una proyección de R, y sea **DEPS** el conjunto de dependencias aplicable a S. Entonces:

- Si (X→Y) está en **DEPS+**, también está en **DEPR+**.
- Para DPs esto no es necesariamente cierto.

Además, al proyectar siempre hay alguna DP que no se transmite completa. En efecto, si X→»Y se transmitiera completa, no se transmitiría completa su complementaria, X→»U − Y. En definitiva, supongamos que la relación R, con dependencias DEPR, se descompone proyectándola en S y T, transmitiendo a éstas las dependencias DEPS y DEPT, respectivamente. Si tomamos dos extensiones válidas de S y T, es decir que cumplan DEPS y DEPT, esto no implica en general que su yunción R = S * T cumpla las dependencias originales DEPR.

Recuérdese, sin embargo, que para dos extensiones cualesquiera de S(A, B) y T(A, C), en su yunción S * T se verifica que A→»B y A→»C.

Supongamos que al obtener S nos damos cuenta, de acuerdo con el significado de los atributos de S, de que existe una DP (X→»Y), aplicable a S y no trivial en S, que no es aplicable a R. Decimos en este caso que (X→»Y) es una DP embebida en R, y explícita en la proyección S de R.

Ejemplo:

Tomemos la relación PREQ siguiente, que indica los alumnos matriculados por asignatura, los prerrequisitos entre éstas y las fechas en que los estudiantes han aprobado los prerrequisitos.

PREQ (AS, AL, PRE, F)

AS: Asignatura en la que el alumno está matriculado
AL: Alumno
PRE: Asignatura prerrequisito
F: Fecha en que aprobó el prerrequisito

Ejemplo de extensión de PREQ:

PREQ	AS	AL	PRE	F
	Fis2	A11	Fis1	1983
	Fis2	A11	Mat1	1983
	Fis2	A12	Fis1	1985
	Fis2	A12	Mat1	1984

Esto significa que, por ejemplo, el Alumno A11 está matriculado en Física-2, aprobó Física-1 en 1983, aprobó Matemáticas-1 en 1983, y que tanto Física-1 como Matemáticas-1 son asignaturas que hay que tener aprobadas antes de cursar Física-2.

La única dependencia es (AL, PRE→F).

Tenemos pues el esquema:

PREQ (AS, AL, PRE, F)

AL PRE→F
Clave: (AS, AL, PRE)
No es FNBC

Es evidente que en PREQ no se cumple la DP (AS→»AL), pues si tomamos dos filas con igual AS, por ejemplo:

Fis2 A11 Fis1 1983
Fis2 A12 Mat1 1984

al intercambiar los alumnos entre ellas se obtienen tuplas que no están en la relación:

Fis2 A12 Fis1 1983
Fis2 A11 Mat1 1984

Esto se debe a que los alumnos pueden haber aprobado los prerrequisitos en años diferentes, como ocurre en el ejemplo.

Descompongamos ahora PREQ para normalizar:

PREQ1 (AL, PRE, F)
AL PRE→F
Clave: (AL, PRE)
FNBC

PREQ2 (AS, AL, PRE)
Clave: (AS, AL, PRE)
FNBC

Aparentemente hemos llegado a un diseño correcto, normalizado y que preserva las dependencias. Sin embargo, de acuerdo con el significado de los atributos, es lógico imponer a las tuplas de PREQ2 la condición (AS↠PRE), y por complementación (AS↠AL), aunque, como ya hemos visto, no se cumplen estas condiciones en PREQ. Es decir, que tanto (AS↠AL) como (AS↠PRE), son DPs embebidas en PREQ, y explícitas en PREQ2.

Si admitimos que AS↠PRE es aplicable a PREQ2, entonces esta relación no es FN4. Normalicémosla:

PREQ3 (AS, PRE)
FN4

PREQ4 (AS, AL)
FN4

Hemos llegado así al esquema de diseño:

PREQ1 (AL, PRE, F);	(Asignaturas aprobadas por los alumnos)
PREQ3 (AS, PRE);	(Prerrequisitos por asignatura)
PREQ4 (AS, AL);	(Alumnos matriculado por asignatura)

En este diseño se debe cumplir una condición de integridad que no es expresable como DF ni DP. Es la siguiente: la proyección de PREQ1 sobre (AL, PRE) debe coincidir con la proyección de (PREQ3 * PREQ4) sobre los mismos atributos.

Con los valores del ejemplo:

PREQ1	AL	PRE	F
	A11	Fis1	1983
	A11	Mat1	1983
	A12	Fis1	1985
	A12	Mat1	1984

PREQ2	AS	PRE
	Fis2	Fis1
	Fis2	Mat1

PREQ4	AS	AL
	Fis2	A11
	Fis2	A12

En resumen, al normalizar hay que estar atento a si aparecen en algún caso, al proyectar, dependencias plurales no explícitas en la relación original, de acuerdo con el significado de los atributos.

4

Vamos a considerar ahora una nueva notación para DPs que puede usarse también para expresar las DPs embebidas.

DEPENDENCIAS JERARQUICAS

Hemos visto que si en una relación R(A, B, C) existe una DP (A→»B), también se verifica que (A→»C). Esto nos sugiere emplear la notación (A→»B | C), para indicar que ambas DPs de verifican. La existencia de estas dos DPs puede también interpretarse como la existencia de dos asociaciones independientes entre los valores de los atributos A y B, por una parte, y los de A y C por otra. Es decir, que a cada valor de A corresponden varios de B y varios de C, y que éstos son independientes de aquéllos.

Ejemplo:

En la relación FABRICA (E, M, P) estudiada anteriormente tenemos que una máquina puede ser usada por varios empleados para fabricar varios productos, y los productos que se pueden fabricar con ella son independientes de los empleados que la usan, es decir, (M→»P | E).

Por otra parte, si volvemos a la relación PREQ (AS, AL, PRE, F), ya estudiada más arriba, vimos ya que las condiciones (AS→»AL) y (AS→»PRE) son DPs embebidas, es decir, que no se verifican en PREQ, pero sí en la proyección de PREQ sobre (AS, AL, PRE). La nueva notación nos permite expresar esto más claramente. En efecto, si decimos que (AS→»AL | PRE) se verifica en la relación PREQ, estamos ya indicando que se trata de dos DPs embebidas, puesto que no se mencionan todos los atributos de PREQ. En resumen, dos asociaciones independientes entre los valores de tres atributos disjuntos, A, B, y C, las representaremos como (A→»B | C), y equivalen a una pareja de DPs, explícitas o embebidas, según que (A, B, C) incluyan o no a todos los atributos de la relación, respectivamente.

Estas ideas pueden generalizarse a más de dos asociaciones. Sea una relación R con atributos (posiblemente compuestos) X, Y_1, Y_2, ..., Y_n, Z, disjuntos. Si existen asociaciones independientes entre los valores de (X, Y_1), (X, Y_2), ..., (X, Y_n), diremos que se verifica una Dependencia Plural Múltiple entre X y (Y_1, Y_2, ..., Y_n). También llamaremos a este tipo de condiciones Dependencias Jerárquicas (DJs), y la representaremos: $(X \twoheadrightarrow Y_1 \mid Y_2 \mid \cdots \mid Y_n)$.

Si $Z = \emptyset$, es decir, si todos los atributos de R están incluidos en (X, Y_1, Y_2, ..., Y_n), se trata de una DJ explícita. En caso contrario diremos que es una DJ embebida o implícita en R, y explícita en la proyección de R sobre (X, Y_1, Y_2, ..., Y_n).

Vamos a repetir la definición de DJ para dar más precisión a la idea de asociaciones «independientes». Para abreviar, vamos a usar las notaciones siguientes, ya definidas anteriormente:

- Proyección de R sobre unos atributos X, Y: R[X, Y]. O sea:

  R[X, Y] = **P**(X, Y) (R)

- Conjunto de todas las parejas de valores de X e Y asociados en R, para un determinado valor, x, del atributo X: **R**(x, Y). O sea:

  $\mathbf{R}(x, Y) = \mathbf{P}(X, Y)\ (\mathbf{S}(X = x)\ (R))$

Definición: Diremos que existe una DJ, $(X \twoheadrightarrow Y_1 \mid Y_2 \mid \cdots \mid Y_n)$, si para todo valor x del atributo X en la relación R se verifica:

$$\mathbf{R}(x, Y_1, Y_2, \ldots, Y_n) = \mathbf{R}(x, Y_1) * \mathbf{R}(x, Y_2) * \cdots * \mathbf{R}(x, Y_n)$$

La propiedad de reversibilidad por yunción al descomponer mediante proyecciones que tenían las DPs es generalizable a las DJs, puesto que aquéllas son un caso particular de éstas. Así, sea una relación R con atributos disjuntos X, Y_1, Y_2, ..., Y_n, que incluyen a todos los atributos de R. La condición necesaria y suficiente para que la descomposición de R en proyecciones sobre (X, Y_1), (X, Y_2), ..., (X, Y_n), sea reversible por yunción es que se verifique $(X \twoheadrightarrow Y_1 \mid Y_2 \mid \cdots \mid Y_n)$.

Vamos a demostrar que esta propiedad es cierta. Para ello utilizaremos las dos identidades siguientes:

1) $R(X, Y_1, Y_2, \ldots, Y_n)$ = (Unión de todas las relaciones de la forma $\mathbf{R}(x, Y_1, Y_2, \ldots, Y_n)$, obtenidas para todos los valores, x, de X en R) = $\cup$ (para todo x) $(\mathbf{R}(x, Y_1, Y_2, \ldots, Y_n))$

2) $\mathbf{R}[X, Y_1] * \mathbf{R}[X, Y_2] * \cdots * R[X, Y_n] =$
   $= \cup$ (para todo x) $(\mathbf{R}(x, Y_1)) * \mathbf{R}(x, Y_2) * \mathbf{R}(x, Y_3) * \cdots * \mathbf{R}(x, Y_n))$

En efecto, veamos que si se cumple $(X \twoheadrightarrow Y_1 \mid Y_2 \mid \cdots \mid Y_n)$, la yunción reproduce la relación R:

$R(X, Y_1, Y_2, \ldots, Y_n) = \cup$ (para todo x) $(\mathbf{R}(x, Y_1, Y_2, \ldots, Y_n)) = \cup$ (para todo x) $(\mathbf{R}(x, Y_1) * \mathbf{R}(x, Y_2) * \cdots * \mathbf{R}(x, Y_n)) = \mathbf{R}[X, Y_1] * \mathbf{R}[X, Y_2] * \cdots * \mathbf{R}[X, Y_n]$.

Recíprocamente, si la yunción de las proyecciones reproduce a R, se cumple que $(X \twoheadrightarrow Y_1 \mid Y_2 \mid \cdots \mid Y_n)$. En efecto:

$R(X, Y_1, Y_2, \ldots, Y_n) = \mathbf{R}[X, Y_1] * \mathbf{R}[X, Y_2] * \cdots * \mathbf{R}[X, Y_n]$; $\cup$ (para todo x) $(\mathbf{R}(x, Y_1, Y_2, \ldots, Y_n)) = \cup$ (para todo x) $(\mathbf{R}(x, Y_1) * \mathbf{R}(x, Y_2) * \cdots * \mathbf{R}(x, Y_n))$. Los dos lados de esta igualdad expresan uniones de relaciones disjuntas. Para que sean iguales, deben serlo término a término. Es decir, para todo x:

$\mathbf{R}(x, Y_1, Y_2, \ldots, Y_n) = \mathbf{R}(x, Y_1) * \mathbf{R}(x, Y_2) * \cdots * \mathbf{R}(x, Y_n)$.

Se ha demostrado que no existe un sistema finito y completo de reglas de inferencia para DJs, aunque existen algunas reglas que pueden ayudarnos a deducir unas DJs de otras en algunos casos. Veamos algunas de estas reglas a título de ejemplo.

1) *Permutación:* Si $(X \twoheadrightarrow Y_1 \mid Y_2 \mid \cdots \mid Y_n)$, también se cumplen las DJs que se obtienen de ésta cambiando el orden de las Ys.

2) *Agrupamiento:* Se pueden agrupar varias Ys en una. Es decir, si $(X \twoheadrightarrow Y_1 \mid Y_2 \mid \cdots \mid Y_n)$, también se cumple, por ejemplo, $(X \twoheadrightarrow Y_1 Y_2 \mid Y_3 \mid \cdots \mid Y_n)$.

3) *Supresión:* Se puede suprimir una Y cualquiera. Es decir, si $(X \twoheadrightarrow Y_1 | Y_2 | \cdots | Y_n)$, también se cumple, por ejemplo, $(X \twoheadrightarrow Y_2 | Y_3 | \cdots | Y_n)$.

4) *Proyección:* Si $(X \twoheadrightarrow Y_1 | Y_2 | \cdots | Y_n)$ y $(Y'_1 \subset Y_1)$, también se cumple que $(X \twoheadrightarrow Y'_1 | Y_2 | Y_3 | \cdots | Y_n)$.

Puesto que una DJ explícita, $(X \twoheadrightarrow Y_1 | Y_2 | \cdots | Y_n)$, equivale a DPs: $X \twoheadrightarrow Y_1$, $X \twoheadrightarrow Y_2$, ..., $X \twoheadrightarrow Y_n$, si X no es una superclave de R, ésta no será FN4, pero se podrá normalizar proyectando sobre $(X\ Y_1)$, $(X\ Y_2)$, ..., $(X\ Y_n)$.

DEPENDENCIAS YUNCIONALES

Sean X_1, X_2, ..., X_n, conjuntos de atributos de U. A la condición de que la descomposición de R proyectando sobre X_1, X_2, ..., X_n, sea reversible por yunción le daremos un nombre específico. La llamaremos Dependencia Yuncional, y la representaremos como $*(X_1, X_2, \ldots, X_n)$.

Dicho de otra forma, en un esquema de relación R se cumple la DY $*(X_1, X_2, \ldots, X_n)$ si todas sus extensiones se pueden reconstruir ayuntando las proyecciones sobre X_1, X_2, ..., X_n. O sea, para todas las extensiones de R se cumple que:

$$R = \mathbf{R}[X_1] * \mathbf{R}[X_2] * \cdots * \mathbf{R}[X_n]$$

Evidentemente, para que se cumpla la condición anterior, todos los atributos de R deben estar contenidos en $X_1 X_2 \ldots X_n$. Es decir, debe ser $(X_1 X_2 \ldots X_n) = U$.

En consecuencia, la condición $*(X_1, X_2)$ puede escribirse también como $(X \twoheadrightarrow Y | Z)$, donde $(X = X_1 \cap X_2)$, $(Y = X_1 - X_2)$ y $(Z = X_2 - X_1)$.

Como ya vimos, la condición $(X \twoheadrightarrow Y | Z)$, con $Z = U - XY$, equivalía a la siguiente: si hay en R dos tuplas con igual X, también existen las que se obtienen intercambiando los valores de Y. Expresemos esto con una notación más precisa. Sean t_1 y t_2 dos tuplas de R. Representemos con $t_1(X)$ el valor del atributo X en la tupla t_1. La condición $(X \twoheadrightarrow Y | Z)$ puede entonces expresarse así:

> **Si existen en R dos tuplas t_1 y t_2 tales que $t_1(X) = t_2(X)$, entonces también existe una tupla t_3, no necesariamnte distinta a t_1 o t_2, tal que $t_3(X) = t_1(X) = t_2(X)$, $t_3(Y) = t_1(Y)$, $t_3(Z) = t_2(Z)$, o lo que es igual: $t_3(XY) = t_1(XY)$ y $t_3(XZ) = t_2(XZ)$.**

Con nuestra notación de DYs esto se expresaría así:

> **Si la condición $*(X_1, X_2)$ se cumple y existen dos tuplas t_1 y t_2 de R tales que $t_1(X_1 \cap X_2) = t_2(X_1 \cap X_2)$, entonces también existe una tupla t_3 en R tal que $t_3(X_1) = t_1(X_1)$ y $t_3(X_2) = t_2(X_2)$.**

Esto se puede generalizar como sigue. Si la condición $*(X_1, X_2, \ldots, X_n)$ se cumple, también se cumple la condición siguiente:

> **Si existen, en R, n tuplas, t_1, t_2, ..., t_n, tales que $t_i(X_i \cap X_j) = t_j(X_i \cap X_j)$, para todo i y j de 1 a n, entonces también existe en R una tupla t, no necesariamente distinta de t_1, t_2, ..., t_n, tal que $t(X_i) = t_i(X_i)$ y $t(X_j) = t_j(X_j)$.**

Ejemplo:

Supongamos que R(A, B, C, D, E) satisface la condición *(ABC, BD, CDE) Tenemos por tanto: X1 = ABC, X2 = BD, X3 = CDE. Supongamos que R contiene las tuplas siguientes, entre otras:

⟨a, b, c, d, e ⟩
⟨a1, b, c, d, e2⟩
⟨a2, b1, c, d, e1⟩

Pongámoslas en este orden:

t_1: ⟨a, b, c, d, e ⟩; $X_1 = ABC$
t_2: ⟨a1, b, c, d, e2⟩; $X_2 = BD$
t_3: ⟨a2, b1, c, d, e1⟩; $X_3 = CDE$

Vemos que:

$t_1(X_1 \cap X_2) = t_2(X_1 \cap X_2) = b$
$t_1(X_1 \cap X_3) = t_3(X_1 \cap X_3) = c$
$t_2(X_2 \cap X_3) = t_3(X_2 \cap X_3) = d$
$t_1(X_1) = \langle a\,b\,c \rangle$
$t_2(X_2) = \langle b\,d \rangle$
$t_3(X_3) = \langle c\,d\,e1 \rangle$

Por tanto, si se cumple la DY *(ABC, BD, CDE) en R, ésta deberá contener la tupla ⟨a, b, c, d, e1⟩.

Si tomamos las tuplas anteriores en otro orden, tendríamos:

t_1: ⟨a1, b, c, d, e2⟩; $X_1 = ABC$
t_2: ⟨a, b, c, d, e ⟩; $X_2 = BD$
t_3: ⟨a2, b1, c, d, e1⟩; $X_3 = CDE$

Vemos que:

$t_1(X_1 \cap X_2) = t_2(X_1 \cap X_2) = b$
$t_1(X_1 \cap X_3) = t_3(X_1 \cap X_3) = c$
$t_2(X_2 \cap X_3) = t_3(X_2 \cap X_3) = d$
$t_1(X_1) = \langle a1\,b\,c \rangle$
$t_2(X_2) = \langle b\,d >$
$t_3(X_3) = \langle c\,d\,e1 \rangle$

Por tanto, si se cumple la DY *(ABC, BD, CDE) en R, ésta deberá contener la tupla ⟨a1, b, c, d, e1⟩.

Como $(X_1, X_2, \ldots, X_n)$ debe contener todos los atributos de R, la condición anterior para DY puede enunciarse de la forma siguiente. Sean $t_1, t_2, \ldots, t_n$, tuplas de R tales que en todas las combinaciones posibles dos a dos de ellas se verifica $t_i(X_i \cap X_j) = t_j(X_i \cap X_j)$. Formemos ahora a una tupla t tal que $t(X_1) = t_1(X_1)$, $t(X_2) = t_2(X_2)$, $t(X_3) = t_3(X_3)$, ..., $t(X_n) = t_n(X_n)$. Una condición necesaria para que R cumpla la DY $*(X_1, X_2, X_3, \ldots, X_n)$ es que, para todo grupo de n tuplas que exista en R con las condiciones anteriores, la tupla t que se obtiene de ellas como se ha indicado también esté en R. En el caso de $n = 2$, esta condición es necesaria y suficiente.

En las definiciones de DY dadas, intrepretaremos que si $(X_i \cap X_j) = \emptyset$, se cumple que $t_i(X_i \cap X_j) = t_j(X_i \cap X_j)$. Si X_1, X_2, ..., X_n, son disjuntos, todas las combinaciones posibles de sus valores existen en R. Por ejemplo, sea R(A, B, C), y sea (a, b) el conjunto de valores de $\mathbb{R}$[A], (1, 2) el de $\mathbb{R}$[B] y (m, n) el de $\mathbb{R}$[C]. Entonces, la siguiente extensión de R cumple la DY $*$(A, B, C):

R	A	B	C
	a	1	m
	a	1	n
	a	2	m
	a	2	n
	b	1	m
	b	1	n
	b	2	m
	b	2	n

Una DY es trivial si se verifica para todas las extensiones posibles de R, sin ninguna restricción. La condición necesaria y suficiente para que la DY $*(X_1, X_2, \ldots, X_n)$ sea trivial es que alguno de sus componentes, tal que X_i, incluya a todos los atributos de R (o sea, que $X_i = U$). En caso de una DP, ya sabemos que toda DP trivial es, o bien de la forma $X \twoheadrightarrow Y$, donde $Y \subseteq X$, o bien de la forma $X \twoheadrightarrow U - X$. El primer caso equivale a la DY $*(XY, XZ)$, con $(Y \subseteq X)$ y $(Z = U - X)$, o sea, $*(X, U)$. El segundo caso es $*(XY, XZ)$, con $(Y = U - X)$ y $(Z = U - XY)$, o sea, $*(U, X)$.

Cuando $(X_1 X_2 X_3 \ldots X_n) = (U') \subset U$, es decir, no incluye a todos los atributos de U, la DY se llama embebida en R, y será por el contrario explícita en la proyección de R sobre U'. Por tanto, esta proyección será descomponible proyectándola sobre $X_1, X_2, \ldots, X_n$.

Existen reglas de inferencia que permiten deducir unas DYs a partir de otras.

Vamos a enunciar a título de ejemplo algunas reglas de inferencia para DYs. Sea un esquema R, con un conjunto U de atributos, y sean $X_1, X_2, \ldots, X_n$, subconjuntos de U tales que $(X_1 X_2 \ldots X_n) = U$.

1) *Permutabilidad.* Si se verifica en una extensión de R la DY $*(X_1, X_2, \ldots, X_n)$, también se verifica cualquier DY que se obtenga de ésta cambiando el orden de las X. Esto es consecuencia de la conmutatividad del operador de yunción.

2) *Reflexividad.* Si consideramos una DY con sólo un componente, éste debe ser U, es decir, se verifica $*(U)$.

3) *Extensibilidad.* Si se verifica en una extensión de R la DY $*(X_1, X_2, \ldots, X_n)$, y X_m es un subconjunto de U, también se cumple $*(X_1, X_2, \ldots, X_n, X_m)$.

4) *Reagrupamiento.* Si se verifica en una extensión de R la DY $*(X_1, X_2, X_3, \ldots, X_n)$, también se cumple $*((X_1 X_2), X_3, \ldots, X_n)$.

6) *Proyección.* Sea S el conjunto de todos los atributos que están en dos o más componentes X_i, es decir, S contiene a $(X_i \cap X_j)$ para todo i, j. Evidentemente, $X_i \cap S$, contiene a todos los atributos de X_i que se repiten en algún X_j. Además, todos los atributos de $U-S$ pertenecen a un solo componente.

Entonces, si se verifica en una extensión de R la DY $*(X_1, X_2, X_3, \ldots, X_n)$, también se cumple la DY embebida $*(X_1 \cap S, X_2 \cap S, \ldots, X_n \cap S)$.

7) *Substitución.* Si se verifica en una extensión de R la DY $*(X_1, X_2, X_3, \ldots, X_n)$ y la DY embebida $*(Y_1, Y_2, \ldots, Y_m)$, y además $X_1 = (Y_1\, Y_2 \ldots Y_m)$, también se verifica la $*(Y_1, Y_2, \ldots, Y_m, X_2, X_3, \ldots, X_n)$.

FORMA NORMAL DE PROYECCION-YUNCION

Del mismo modo que las anomalías producidas por DPs desaparecían en la cuarta Forma Normal (FN4), las producidas por DYs pueden eliminarse transformando a una forma normal que suele designarse como quinta Forma Normal (FN5), o también Forma Normal de Proyección-Yunción (FNPY).

Recordemos las anomalías de actualización debidas a las DPs. Decíamos que si (A→»B) y A no es una superclave se pueden presentar anomalías de actualización. Dicho de otra forma, si A es una superclave, cuando actualizamos una extensión de R basta con tener en cuenta que no se viole la condición de que A es una superclave. No hace falta ninguna otra comprobación. Así, al añadir una nueva fila sólo hay que comprobar que el valor que tiene A no existe ya en alguna de las filas existentes. Una vez cumplida esta condición, ya no hay que comprobar más, pues A→»B se cumplirá por añadidura. Es decir, diremos que no hay anomalías, y que un esquema de relación R está normalizado si todas las condiciones de integridad aplicables al esquema son inferibles a partir de sus claves.

En resumen, definiremos la FN5 como sigue:

Sea un esquema de relación R, con un conjunto DEP de DFs y DYs. Diremos que R está en Forma Normal de Proyección-Yunción (FNPY o FN5) si todas las DFs y DYs no triviales aplicables al esquema R, son inferibles a partir de las claves de R.

Una condición necesaria para que $*(X_1, X_2, \ldots, X_n)$ sea inferible de las claves, es que haya al menos una pareja de componentes cuya intersección, $X_i \cap X_j$, sea una superclave. Para $n=2$ esta condición es necesaria y suficiente.

Comprobemos esto.

En efecto, supongamos que $*(X_1, X_2, \ldots, X_n)$ es aplicable a R y que ninguna combinación $X_i \cap X_j$ de esta DY es superclave (o sea, que para todo i, j, o bien la intersección $(X_i \cap X_j)$ es vacía ($\emptyset$), o bien, si no es vacía, no contiene una clave). Representaremos con X_{ij} a la intersección $(X_i \cap X_j)$.

Sea una extensión de R formada por n tuplas, $t_1, t_2, \ldots, t_n$, construidas de la forma siguiente:

1) t_1 tiene valores cualesquiera.

2) t_2 tiene en X_{12} los mismos valores que t_1 (si X_{12} no es $\emptyset$), y el resto son todos diferentes,uno a uno, a los de t_1.

3) t_3 tiene en X_{13} los mismos valores que t_1, y en X_{23} los mismos que t_2, y en el resto diferentes, uno a uno, que los de t_1 y t_2.

4) Etc.

Este grupo de n tuplas cumple dos a dos que $t_i(X_{ij}) = t_j(X_{ij})$, y todos los demás valores son diferentes, por lo que cumple que los atributos de las claves no toman valores repetidos. Por otra parte, la tupla t, tal que $t(X_i) = t_i(X_i)$ para todo i, no está en esta extensión de R, luego no se cumple en ésta la $*(X_1, X_2, \ldots, X_n)$. Es decir, hemos construido una extensión de R que satisface la unicidad de las claves, pero no la DY, por lo tanto ésta no es inferible de las claves. Dicho de otra forma, para toda DY inferible de las claves es imposible que todas sus X_{ij} sean no superclaves. O, lo que es igual, para toda DY inferible de las claves, alguna X_{ij} es no vacía y es una superclave.

Esta propiedad permite construir un algoritmo para determinar si una DY dada es inferible de las claves. El algoritmo, debido a Fagin, se basa en aplicar iterativamente esta propiedad y la de reagrupamiento. Si consideramos una DY $*(X_1, X_2, \ldots, X_n)$ que sea inferible de las claves, tendrá una pareja de componentes cuya intersección sea superclave. Sean X_1 y X_2 estos componentes. Aplicando la regla de reagrupamiento se deduce la DY $*((X_1 X_2), X_3, \ldots, X_n)$ que, puesto que se infiere de la DY dada, también es inferible de las claves, y se le puede aplicar el mismo proceso. Repitiendo este proceso se llegará a la DY $*(X_1 X_2 \ldots X_n)$, es decir, $*(U)$. Por tanto, si la DY dada es inferible de las claves, al ir reagrupando se llega a $*(U)$. Puede demostrarse que el recíproco también es cierto. En definitiva, sí se llega a $*(U)$ la DY es inferible de las claves y en caso contrario no lo es.

Obviamente, si R está en FN5 también está en FN4, pero no necesariamente al revés.

FORMAS NORMALES EN GENERAL

En cierto sentido, la FN5 es la forma normal «definitiva». No puede haber más tipos de formas normales sobre condiciones de integridad basadas en operaciones de (Proyección-Yunción). Sin embargo, el camino seguido para definir la FN5 se puede generalizar a otros tipos de condiciones de integridad Antes de hacerlo, vamos a justificarlo, viendo que es aplicable asimismo a las definiciones de FNBC y FN4.

Consideremos primero la definición de FNBC. Dijimos que un esquema de relación, R, está en FNBC si para toda DF no trivial de **DF+**, el antecedente es una superclave. Vamos a justificar que esto es equivalente a decir que esta DF es inferible de las claves.

4

Sea un esquema R y un conjunto de DFs, **DF**, aplicables a R. Sean K_1, K_2, ..., K_m, las claves de R.

1) Si R está en FNBC, todas sus DFs no triviales son inferibles de las claves.

   En efecto, si R está en FNBC, en toda DF no trivial, tal que $X \rightarrow Y$, aplicable a R, se cumple que X contiene a una clave al menos, por ejemplo a K_1, es decir, $X \supseteq K_1$. Por tanto, $X \rightarrow K_1$, y como $K_1 \rightarrow Y$, resulta que $X \rightarrow Y$. Es decir, toda DF es inferible a partir de las claves.

2) Si todas las DFs son inferibles de las claves, R está en FNBC.

   En efecto, sea $X \rightarrow Y$ una DF no trivial aplicable a R en la que X no es una superclave. Consideremos una extensión de R formada por las tuplas:

   $\langle x\, y_1\, z_1 \rangle$,
   $\langle x\, y_2\, z_2 \rangle$,

   (donde todos los elementos de y_1 son diferentes a los de y_2, y los de z_1 a los de z_2).

   Esta extensión satisface las claves porque, al no ser X una superclave, todas las claves están contenidas en (YZ). Pero como $\langle y_1\, z_1 \rangle$ y $\langle y_2\, z_2 \rangle$ son distintos en todos los atributos, las claves no se repiten. En definitiva, esta extensión de R satisface las condiciones de las claves. Sin embargo, no cumple $X \rightarrow Y$. Luego esta DF no es inferible de las claves (pues una condición de integridad es inferible de otras, por definición, cuando toda extensión que satisface a éstas también satisface a aquélla).

   En resumen, si todas las DFs no triviales aplicables a R son inferibles de las claves, sus antecedentes son superclaves, y por tanto R está en FNBC.

Veamos ahora la definición de FN4. Dijimos que un esquema de relación, R, está en FN4 si para toda DP no trivial de **DP**+ el antecedente es una superclave. Vamos a justificar que esto es equivalente a decir que esta DP es inferible de las claves, para lo cual seguiremos el mismo método que más arriba.

Sea un esquema R y un conjunto de DFs y DPs, aplicables a R. Sean K_1, K_2, ..., K_m, las claves de R.

1) Si R está en FN4, todas sus DPs no triviales son inferibles de las claves.

   En efecto, si R está en FN4, en toda DP no trivial, tal que $X \twoheadrightarrow Y$, aplicable a R, se cumple que X contiene a una clave al menos, por ejemplo a K_1, es decir, $X \supseteq K_1$. Por tanto, $X \rightarrow K_1$, y como $K_1 \rightarrow Y$, resulta que $X \rightarrow Y$, y en consecuencia, $X \twoheadrightarrow Y$. Es decir, toda DP es inferible a partir de las claves.

2) Si todas las DPs son inferibles de las claves, R está en FN4.

En efecto, sea X→→Y una DP no trivial aplicable a R en la que X no es una superclave. Consideremos una extensión de R formada por las tuplas:

$\langle x\, y_1\, z_1 \rangle$
$\langle x\, y_2\, z_2 \rangle$,

(donde todos los elementos de y_1 son diferentes a los de y_2, y los de z_1 a los de z_2).

Esta extensión satisface las claves, como ya vimos anteriormente. Sin embargo, no cumple X→→Y. Luego esta DP no es inferible de las claves.

En resumen, si todas las DPs no triviales aplicables a R son inferibles de las claves, sus antecedentes son superclaves.

En general, dado un tipo, CI, de condiciones de integridad cualesquiera, podemos definir una Forma Normal para las condiciones CI, y la representaremos FN-CI, de la forma siguiente:

Un esquema R de la relación está en FN-CI si todas las condiciones de tipo CI no triviales aplicables a R son inferibles de las claves de R.

Así es para las Formas Normales definidas hasta ahora:

Tipo de CIs	*Forma Normal*
DFs	FNBC
DPs y DFs	FN4
DYs y DFs	FNPY

Por tanto, si un esquema de diseño está normalizado, el Sistema de Gestión de Bases de Datos (SGBD) sólo necesita comprobar, cada vez que se actualiza una relación, que los atributos que son claves no toman valores repetidos. No necesita hacer comprobaciones adicionales, pues todas las restantes condiciones de integridad se cumplirán automáticamente por ser consecuencia de las claves. En definitiva, el SGBD sólo necesita estar dotado de mecanismos que le permitan asegurar la unicidad de los valores de las claves.

NORMALIZAR O DESNORMALIZAR

Dado un esquema de diseño formado por varios esquemas de relación, R_1, R_2, ..., R_n, y un conjunto de dependencias, habrá en general algunos de ellos normalizados y otros no, y los normalizados lo estarán a distinto grado (FN3, FNBC, etc.). Podemos preguntarnos si es conveniente normalizar todos los esquemas y hasta qué grado. Esta pregunta debe responderla el analista o diseñador en cada caso particular, aunque deberá tener en cuenta algunas consideraciones de tipo general como las siguientes:

- Cuanto mayor sea la normalización más fielmente se representa el mundo real en el esquema y, por tanto, el diseño será probablemente más estable.

- Si se normaliza al máximo, ni el sistema ni los programas deben responsabilizarse, para proteger la integridad de los datos, de otra cosa que de la unicidad de valores de las claves. Esto disminuye los riesgos de integridad y la redundancia, y por consiguiente mejora el rendimiento en operaciones de actualización.
- Por el contrario, una mayor normalización suele implicar un número también mayor de relaciones (más pequeñas, o sea, con menos atributos cada una). Esto puede obligar a un incremento en el número de operaciones de yunción, con lo que el rendimiento en consultas puede ser peor.

En conclusión, es conveniente normalizar, como directriz general de diseño, aunque puede haber excepciones por razones de rendimiento o facilidad de uso en consultas.

DEPENDENCIAS GENERALIZADAS

Vamos a introducir una nueva notación para expresar las DFs, DPs y DYs que nos permitirá generalizar el concepto de dependencia.

Recordemos la definición de DF. Si tenemos un esquema de relación, por ejemplo R(A, B, C, D), decimos que R cumple la DF A→B si para toda extensión válida de R se verifica que si dos tuplas tienen iguales valores en A, también son iguales sus valores en B. O, dicho de otra forma, si hay en R dos tuplas tales como ⟨a b1 c1 d1⟩ y ⟨a b2 c2 d2⟩, debe ser b1 = b2. Esto lo podemos representar con una notación tabular de la forma siguiente:

```
a    b1    c1    d1
a    b2    c2    d2
---------------------
       b1 = b2
```

Las filas por encima de la raya se llaman hipótesis, y la que está debajo se llama conclusión.

Consideremos ahora la condición A↠B. Si se cumple, quiere decir que si hay en R dos tuplas con iguales valores en A, también están en R las que se obtienen intercambiando los valores de B. Es decir, si hay en R dos tuplas tales como ⟨a b1 c1 d1⟩ y ⟨a b2 c2 d2⟩, también debe estar en R la tupla ⟨a b1 c2 d2⟩ (no hace falta expresar que también esté la ⟨a b2 c1 d1⟩, pues ésta se deduce de la condición simplemente considerando las tuplas de partida en distinto orden). Esto lo podemos expresar como antes en forma de dos hipótesis y una conclusión:

```
a    b1    c1    d1
a    b2    c2    d2
---------------------
a    b1    c2    d2
```

Pueden dejarse en blanco los valores que sólo figuran una vez. Por ejemplo, la notación anterior para expresar A↠B podría quedar así:

A	B	C	D
a	b	—	—
a	—	c	d
a	b	c	d

Consideremos ahora la DY $*(AB, BC, CD)$. Si esta DY se cumple en R, quiere decir que para todo grupo de 3 tuplas de R de la forma ⟨a b c1 d1⟩, ⟨a2 b c d2⟩ y ⟨a3 b3 c d⟩, también debe existir en R la tupla ⟨a b c d⟩. Con nuestra notación tabular:

A	B	C	D
a	b	—	—
—	b	c	—
—	—	c	d
a	b	c	d

Por convenio, un símbolo no puede aparecer más que en una columna (tanto si es un símbolo de las hipótesis como de la conclusión), pero puede repetirse dentro de su columna. Puede haber símbolos en la conclusión que no aparezcan en las hipótesis. Estos símbolos los llamaremos singulares.

Vemos que hay dos tipos de conclusiones: o bien una igualdad entre símbolos no singulares, o bien una tupla.

A las condiciones de integridad expresadas mediante esta notación, las llamaremos Dependencias Generalizadas (DGs). Si todos los símbolos de la conclusión aparecen en las hipótesis (es decir, si no hay símbolos singulares), diremos que es una Dependencia explícita. En caso contrario diremos que es implícita o embebida. Así, por ejemplo, en el esquema R(A, B, C, D), la DP embebida $A \twoheadrightarrow B \mid C$, se representa como:

A	B	C	D
a	b	c1	d1
a	b2	c	d2
a	b	c	d3

O también:

A	B	C	D
a	b	—	—
a	—	c	—
a	b	c	—

Diremos que una extensión de R cumple una DG dada si, para toda substitución de los símbolos por valores tales que las hipótesis sean tuplas existentes en R, se verifica la igualdad expresada en la conclusión (si ésta es una

igualdad), o se verifica que la tupla de la conclusión también existe en R (es decir, que existe en R una tupla que coincide con la conclusión en los símbolos no singulares).

Ejemplo:

Sea la extensión R:

R	A	B	C
	0	1	2
	0	3	4
	0	3	2
	5	1	4

Y sea la DG embebida:

A	B	C
a	b	c1
a	b2	c
a3	b	c

Vamos a comprobar que la relación R anterior cumple esta DG. Para ello tenemos que ver que para toda substitución posible de valores de a, b, c, a3, b2 y c1, tal que las hipótesis estén en R, también la conclusión está en R.

Consideremos primero una substitución de valores tal que las dos hipótesis produzcan la misma tupla de R. Por ejemplo, a = 0, b = 1, c1 = 2, b2 = 1, c = 2; la conclusión será ⟨a3 1 2⟩; vemos que esta tupla existe en R para a3 = 0. En general, si ⟨a b c1⟩ y ⟨a b2 c⟩ representan la misma tupla de R, deberá ser b2 = b y c1 = c, con lo que ⟨a b c⟩ estará en R. Además la conclusión será ⟨a3 b c⟩, que evidentemente, si a3 = a, existe en R.

Consideremos ahora substituciones tales que las hipótesis correspondan a dos tuplas diferentes de R. Como ambas tuplas tienen el mismo valor de A, deberá ser a = 0. Hay, pues, tres posibles substituciones:

Substitución	a	b	c1	b2	c	Conclusión	a3
1	0	1	2	3	4	⟨a3 1 4⟩	5
2	0	1	2	3	2	<a3 1 2>	0
3	0	3	4	3	2	⟨a3 3 2⟩	0

Vemos pues que se cumple en R la DG mencionada.

Análogamente a como hicimos con los otros tipos de dependencias, podríamos definir que un esquema de relación está normalizado para DGs si todas las DGs no triviales son inferibles a partir de las claves. Sin embargo, este concepto no suele utilizarse, pues es difícil de manejar. La mayor utilidad de las DGs es que sirven de base a un algoritmo para deducir si una DG dada es inferible de otras, al que llamaremos algoritmo de Batida.

Antes de pasar a describir este algoritmo vamos a definir una operación que llamaremos «aplicación» de una DG a una relación. Sea d una DG y sea R una extensión de un esquema de relación. Supongamos que encontramos una substitución de los símbolos de las hipótesis de d por valores tales que todas ellas resultan ser tuplas existentes en R. Si la conclusión de d es una igualdad, la aplicación de d a R consiste en igualar en R todos los valores asociados a los símbolos que se igualan en la conclusión. Para ello, basta subtituir uno cualquiera de ellos por el otro en todos los de R en que aparezca.

Si la conclusión de d es una tupla que no tiene símbolos singulares, la aplicación de d a R consiste en añadir a R la tupla obtenida al substituir todos los símbolos de la conclusión por sus valores correspondientes.

Si la conclusión de d es una tupla con símbolos singulares, la aplicación de d a R consiste en añadir a R la tupla obtenida substituyendo en la conclusión los símbolos no singulares por sus correspondientes valores, y los singulares por valores cualesquiera que no existan en otras tuplas de R. Si estos valores pudieran subtituirse por otros ya existentes en R de manera que la nueva tupla ya existiera completa en R, no se añadirá nada a R.

Ejemplo 1:

Aplicación de una DF a una extensión R.

R	A	B	C
	1	2	3
	1	4	5
	2	4	3

d	A	B	C
	a	b1	c1
	a	b2	c2
		b1 = b2	

Substitución de valores: a = 1, b1 = 2, c1 = 3, b2 = 4, c2 = 5.

Resultado de aplicar d a R para esta substitución de valores: b2 = b1 = 2.

Queda:

R'	A	B	C
	1	2	3
	1	2	5
	2	4	3

Ejemplo 2:

Aplicación de una DP a una extensión R.

R	A	B	C
	1	2	3
	1	4	5
	2	4	3

d	A	B	C
	a	b1	c1
	a	b2	c2
	a	b1	c2

Substitución de valores: a = 1, b1 = 2, c1 = 3, b2 = 4, c2 = 5. Tupla de conclusión: ⟨1 2 5⟩.

Resultado de aplicar d a R para esta substitución de valores:

R′	A	B	C
	1	2	3
	1	4	5
	2	4	3
	1	2	5

Si consideramos ahora la substitución de valores: a = 1, b1 = 4, c1 = 5, b2 = 2, c2 = 3, la tupla de conclusión será ⟨1 4 3⟩, y el resultado de aplicar d a R′ será:

R″	A	B	C
	1	2	3
	1	4	5
	2	4	3
	1	2	5
	1	4	3

Cualquier otra substitución de valores que consideremos no nos permitirá ya aplicar d a R″ de forma que R″ cambie. Esto significa, con otras palabras, que para todas las substituciones posibles de valores tales que las hipótesis de d pertenezcan a R″, la conclusión de d también existe en R″. O, lo que es igual, R″ satisface la condición expresada por la DG d.

En general, la aplicación reiterativa de una DG, d, a una relación R, hasta que la relación ya no cambie, transforma a R en otra relación en la que se verifica la DG d.

Ejemplo 3:

Aplicación de una DP embebida:

R	A	B	C	D
	1	2	3	6
	1	4	5	8

d	A	B	C	D
	a	b1	c1	d1
	a	b2	c2	d2
	a	b1	c2	d3

Substitución: a = 1, b1 = 2, c1 = 3, d1 = 6, b2 = 4, c2 = 5, d2 = 8. Tupla de conclusión: ⟨1 2 5 x⟩.

Aplicación de d a R:

R′	A	B	C	D
	1	2	3	6
	1	4	5	8
	1	2	5	x

Consideremos otra substitución: a = 1, b1 = 4, c1 = 5, d1 = 8, b2 = 2, c2 = 3, d2 = 6. La tupla de conclusión será ⟨1 4 3 y⟩, y el resultado de aplicar d a R′ será:

R″	A	B	C	D
	1	2	3	6
	1	4	5	8
	1	2	5	x
	1	4	3	y

Cualquier otra substitución no hace ya cambiar a R″. Consideremos, por ejemplo: a = 1, b1 = 4, c1 = 5, d1 = 8, b2 = 2, c2 = 5, d2 = x. La tupla de conclusión será: ⟨1 4 5 z⟩. Vemos que para z = 8 se obtiene una tupla ya existente. Por tanto, no se añade nada a R″ y ésta queda igual.

LA BATIDA

Vamos a describir aquí el algoritmo para averiguar si una DG dada es consecuencia de otras. Lo llamaremos Algoritmo de Batida («Chase Algorithm»).

Supongamos que tenemos un esquema de relación R en el que se verifica un conjunto, **DG**, de Dependencias Generalizadas:

DG = $(d_1, d_2, d_3, \ldots, d_n)$

El Algoritmo de Batida tiene por objetivo averiguar si otra Dependencia Generalizada dada, d, es inferible del conjunto **DG**.

Como ya hemos dicho anteriormente, d es inferible de **DG** si, y sólo si, todas las extensiones de R que cumplan **DG**, cumplen también la DG d.

Veamos una descripción intuitiva de la idea en que se basa el algoritmo. De acuerdo con la notación empleada para una DG, si consideramos sólo las tuplas hipótesis de d, nos queda un conjunto de tuplas formadas con símbolos que puede considerarse como una relación, r, que es una extensión del esquema R.

Tomemos la DG d_1 y apliquémosla a r reiteradamente hasta obtener una relación, r_1, que ya no cambie más. Evidentemente, r_1 satisface la condición expresada por la DG d_1. Apliquemos ahora $(d_2, d_3, \ldots, d_n, d_1)$ repetidamente hasta llegar a una relación, r_n, que ya no cambie más. Esta relación satisface al conjunto de condiciones **DG.**

Supongamos que la conclusión de d es una tupla, t, que no está en r_n. Dicho de otra forma, r_n es una extensión de R que satisface todas las **DG**, pero no satisface la condición d. Por tanto, d no es inferible de **DG**. Por el contrario, puede demostrarse que si t está en r_n, r_n es una extensión de R que cumple tanto **DG** como d.

Basándose en estas ideas se construye el Algoritmo de Batida, que consiste en lo siguiente.

Algoritmo de Batida

1) Considerar la extensión, r, de R, formada por las hipótesis de d.
2) Aplicarle las Dependencias de **DG**, en cualquier orden, hasta que:
   2.1) O bien ya no cambia más.
   2.2) O bien, si la conclusión de d es una tupla sin símbolos singulares, esta tupla existe en r, o bien, si la conclusión tiene algunos símbolos singulares, existe en r alguna tupla que coincide con t en todos sus símbolos no singulares.
   2.3) O bien, si la conclusión de d es una igualdad de valores, éstos se han igualado ya en r.
3) Si nos paramos en el punto 2.1) anterior, d no es inferible de **GD**. Si nos paramos en los puntos 2.2) ó 2.3) anteriores, sí es inferible.

Este procedimiento es finito si todas las dependencias de **DG** son explícitas, aunque crece exponencialmente con el número de dependencias. Además, si en **DG** hay alguna dependencia embebida, no es seguro que sea finito (es decir, no es un algoritmo). Sin embargo, si aún en este caso llega a pararse en algún momento, el resultado que produce es válido. Si **DG** está formado por DFs y DPs exclusivamente es más eficiente aplicar algoritmos basados en las reglas de inferencia que el Algoritmo de Batida. No hay algoritmo conocido que permita resolver este problema si **DG** incluye, por ejemplo, DPs embebidas.

Veamos unos ejemplos de aplicación del Algoritmo de Batida.

Ejemplo 1:

Sea el esquema R(A, B, C, D), en el que se cumplen las DGs:

DG:

A→→B | C
B→D

Sea d: A→→C | D

Vamos a demostrar que d es inferible de **DG**.

Expresemos las DGs en notación tabular:

A→→B | C:	a1	b1	c1	d1
	a1	b2	c2	d2
	a1	b1	c2	d3
B→D:	a4	b4	c4	d4
	a5	b4	c5	d5
		d4=d5		
A→→C | D:	a6	b6	c6	d6
	a6	b7	c7	d7
	a6	b8	c6	d7

Obtenemos r prescindiendo de la conclusión de d:

r	A	B	C	D
	a6	b6	c6	d6
	a6	b7	c7	d7

Apliquemos a r la DG A→→B | C:

r′	A	B	C	D
	a6	b6	c6	d6
	a6	b7	c7	d7
	a6	b7	c6	d8

Apliquemos a r′ la DF B→D:

r″	A	B	C	D
	a6	b6	c6	d6
	a6	b7	c7	d7
	a6	b7	c6	d7

Vemos que la tupla de la conclusión de d, ⟨a6 b8 c6 d7⟩, coincide con la ⟨a6 b7 c6 d7⟩, que está en r″, en todos los símbolos menos el que es singular (b8). Por tanto, d es inferible de **DG**.

Ejemplo 2:

Vamos a usar el Algoritmo de Batida para demostrar que si todas las DPs aplicables a R son inferibles de sus claves, entonces R es FN4 (ya hemos demostrado esto anteriormente).

Sea, en efecto, una DP cualquiera, X→→Y, aplicable a R, no trivial, con X e Y disjuntos, y en la que X no es una superclave. Vamos a demostrar que, en este caso, X→→Y no es inferible de las claves.

Sean K_1, K_2. ..., K_n, las claves de R, o sea que se cumple:

$K_1 \rightarrow U$, $K_2 \rightarrow U$, ..., $K_n \rightarrow U$

Sea $Z = U - XY$. Como X no es una superclave, todas las claves tienen que tener algún atributo en YZ, es decir, para todo i, $K_i \cap YZ = \emptyset$.

Expresemos X→→Y en notación tabular:

X	Y	Z
x_1	y_1	z_1
x_1	y_2	z_2
x_1	y_2	z_1

4

Partimos de:

r	X	Y	Z
	x_1	y_1	z_1
	x_1	y_2	z_2

Apliquemos la DF $K_1 \rightarrow U$ a r. Como K_1 tiene algún componente en (YZ) y ($y_1 z_1$) tiene todos sus elementos distintos a ($y_2 z_2$), la aplicación de $K_1 \rightarrow U$ no produce ningún cambio en r. Lo mismo pasa para $K_2 \rightarrow U$, ..., $K_n \rightarrow U$. Por tanto, el algoritmo termina, y vemos que el resultado, r, no incluye ninguna tupla igual a la conclusión de $X \twoheadrightarrow Y$. Luego esta DP no es inferible de las claves. Esto significa que si todas las DPs son inferibles de las claves, R tiene que estar en FN4, pues si así no fuera, habría alguna $X \twoheadrightarrow Y$ en la que X no sería superclave, y por tanto no podría ser inferible de las claves, lo que contradice la hipótesis de partida.

EJERCICIOS PROPUESTOS

1) *Sea el esquema R(A, B, C, D, E) con las dependencias:*

A→BC; BC→A; BCD→E; E→C

Normalizar R en FNBC.

2) *Sean los atributos A (Agente), O (Oficina), I (Inversor), V (Clase de valor), N (Nominal) y D (Dividendo), y el esquema de relación R(A, O, I, V, N, D), con las siguientes dependencias:*

V→D; I→A; IV→N; A→O

Normalizar R en FNBC.

3) *Sea el esquema R(A, B, C, D) con las dependencias:*

AB→D; C→D; AB→C; C→B

a) Normalizar R en FN3. ¿Se preservan las dependencias?

b) Normalizar R en FNBC. ¿Se preservan las dependencias?

c) Aplicar a R el algoritmo de descomposición que preserva dependencias. Comprobar que se obtienen relaciones FN3.

4) *Sea el esquema R(A, B, C, D, E) con las dependencias:*

A→BCDE; CD→E; CE→B.

Normalizar R en FNBC. ¿Se preservan las dependencias?

5) *Sea el esquema R(A, B, C, D, E, F) con las dependencias:*

$DE \rightarrow C$; $C \rightarrow F$; $F \rightarrow D$; $CE \rightarrow A$; $EF \rightarrow B$

a) Normalizar R en FNBC. ¿Se preservan las dependencias?

b) Normalizar R en FN3. ¿Se preservan las dependencias?

6) *Sea el esquema R(A, B, C, D, E) con las dependencias:*

$C \rightarrow A$; $A \rightarrow BC$; $BCD \rightarrow E$; $E \rightarrow C$

a) Normalizar R en FNBC. ¿Se preservan las dependencias?

b) Normalizar R en FN3. ¿Se preservan las dependencias?

c) Aplicar a R el algoritmo de descomposición que preserva dependencias. Comprobar que se obtienen relaciones FN3.

7) *Añadir a la relación siguiente las filas necesarias para que se cumplan en ella las DPs: $A \twoheadrightarrow BC$; $CD \twoheadrightarrow BE$:*

A	B	C	D	E
a	b	c	d	e
a	1	c	2	e
3	b	c	d	4

8) *Sea el esquema R(A, B, C, D, E) con las dependencias:*

$A \twoheadrightarrow BC$; $DE \twoheadrightarrow C$

Demostrar que $AD \twoheadrightarrow BE$

9) *En el esquema del ejercicio 2 anterior queremos guardar la historia de los dividendos de cada tipo de valor. Entonces, en vez de cumplirse $V \rightarrow D$, tendremos $V \twoheadrightarrow D$.*

Normalizar R en FN4. ¿Se preservan las dependencias?

10) *Sea el esquema R(A, B, C, D, E, F) con las dependencias:*

$A \twoheadrightarrow BCD$; $B \rightarrow AC$; $C \rightarrow D$

Normalizar R en FN4. ¿Se preservan las dependencias?

11) *Sea el esquema R(A, B, C, D, E) con las dependencias:*

$A \twoheadrightarrow BC$; $D \rightarrow C$

a) Demostrar que $A \rightarrow C$.

b) Normalizar R en FN4. ¿Se pierden dependencias?

12) *Sea la siguiente extensión de R(A, B, C):*

A	B	C
a	1	A
a	2	B
b	3	C
c	3	D
d	4	E
e	5	E

a) ¿Satisface esta extensión de R la DY: *(AB, AC, BC)?

b) Añadimos a esta extensión la fila ⟨f 1 B⟩. Añadir las filas que sean necesarias para que se siga cumpliendo la DY anterior.

13) *Sea el esquema R(A, B, C, D, E) con las dependencias:*

AB→C; C→E; E→C; C→D; AB→E

a) ¿Está R en FNBC?

b) Descomponemos R por proyecciones en R1(A, B, C), R2(A, D, E) y R3(C, E). ¿Es esta descomposición reversible por yunción?

c) ¿Preserva las dependencias?

d) Normalizar R1, R2 y R3 en FNBC.

14) *Sea el esquema R(A, B, C, D, E) con las dependencias:*

A→C; B→C; C→D; DE→C; CE→A; *(AD, AB, BE, CDE, AE)

a) Hallar las claves.

b) ¿Es la DY inferible de las DFs?

c) ¿Es la DY inferible de las claves?

d) ¿Está R en FN5?

15) *Sea el esquema R(A, B, C, D, E) con las claves:*

K_1: (BCE); K_2: (ABE)

Decir si son inferibles de las claves las DYs siguientes:

a) *(ABCE, BCDE).

b) *(ABCE, BDE, ACD).

c) *(ABCE, BCE, ABDE).

16) *En el esquema del ejercicio 2 anterior descomponemos R por proyecciones en R1(V, D), R2(I, A), R3(I, V, D) y R4(A, O). ¿Es esta descomposición reversible por yunción?*

17) *En el esquema del ejercicio 2 anterior descomponemos R por proyecciones en R1(I, V, N), R2(I, A), R3(V, D) y R4(I, V, O). ¿Es esta descomposición reversible por yunción? ¿Preserva las dependencias?*

18) *Sea el esquema R(A, B, C, D, E). Los atributos (ABC) y (ABD) toman valores que no se repiten (es decir, son claves o superclaves). Además, se verifica AB→→C. Hallar una clave de R.*

19) *Sea el esquema de relación MATRIM (H, M, S, FB, FD), con el significado siguiente:*

En MATRIM hay una fila por cada matrimonio celebrado en el país desde 1970.

Atributo H: Nombre del marido. Se supone que no hay personas diferentes con igual nombre.

Atributo M: Nombre de la mujer.

Atributo S: Situación del matrimonio. Puede ser una de las siguientes:

- C: Casados todavía.
- D: Ya divorciados.
- S: Separados.
- F: Alguno de los cónyuges, o ambos, han fallecido.

Atributo FB: Fecha de la boda.

Atributo FD: Fecha de disolución del matrimonio, bien por separación o divorcio, bien por fallecimiento de uno o ambos cónyuges.

Enunciar las condiciones de integridad que lógicamente deberán cumplir todas las extensiones válidas de MATRIM.

5 Diagramas para diseño

DIAGRAMAS PARA AYUDAR EN EL DISEÑO

Para diseñar un esquema de BD podríamos partir de un conjunto inicial de esquemas de relación, expresar las dependencias que se presentan entre sus atributos y someter este esquema a un proceso de normalización, según se ha visto en anteriores capítulos. Sin embargo, este proceso no proporciona criterios para formular, a priori, el conjunto inicial de esquemas más adecuado para llegar a un buen diseño.

Además, según hemos visto, los algoritmos de normalización pueden ser de ejecución costosa y, según el orden seguido en el proceso, puede llegarse a distintos resultados. Todo ello hace que sea útil en la práctica emplear métodos más intuitivos, apoyados en representaciones gráficas, que ayuden a expresar y comprender más fácilmente las interrelaciones entre los datos y apoyen el proceso de normalización.

En el presente capítulo y en el siguiente vamos a proponer un método gráfico, basado en una mezcla de los modelos de datos binario y de Entidad/Asociación. La finalidad de combinar ambos modelos es manejar un concepto de entidad que sea lo más objetivo posible. En un modelo de Entidad/Asociación (Entity/-Relationship) hay que definir las entidades que se van a considerar en el modelo, lo que suele plantear dificultades para los usuarios que participen en el diseño y que no siempre están habituados a manejar estos conceptos. En el proceso propuesto, por el contrario, no es necesario predefinir las entidades, pues éstas se pueden obtener como un subproducto del proceso, sirviendo éste de apoyo y guía a la experiencia e intuición del diseñador. Sin embargo, el proceso también es aplicable si se le dan predefinidas algunas entidades, o todas, lo que permite acortar el trabajo a los diseñadores con experiencia.

El método parte de la definición de conjuntos de datos, elementales o compuestos, y de su representación en un diagrama. A éste se le aplican unas transformaciones que permiten ir obteniendo relaciones más complejas, por lo que podríamos decir que se sigue un metódo de síntesis. Por el contrario, los procesos de normalización descritos en el capítulo anterior siguen un método

analítico en sentido inverso, partiendo de relaciones complejas y descomponiéndolas en otras más simples.

Para transformar el diagrama hay que analizar antes algunas propiedades de algunos tipos de condiciones de integridad entre relaciones, que completen a las condiciones de integridad intrarrelación estudiadas en capítulos precedentes (dependencias funcionales, plurales, etc.). A esto se dedica parte de la materia del presente capítulo, en el que también vamos a describir cómo construir y transformar estos diagramas hasta llegar a definir las entidades y el esquema de diseño.

Este capítulo está estructurado gradualmente, introduciendo cada nuevo concepto después de plantear el problema que se pretende resolver con él. Esto puede hacer la presentación algo redundante. El lector que desee ganar tiempo en una segunda lectura o repaso puede centrarse en los párrafos siguientes:

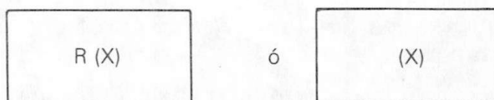
- Representación de conjuntos de valores.
- Representación de asociaciones binarias implícitas.
- Asociaciones implícitas entre relaciones unarias.
- Asociaciones implícitas entre relaciones n-arias.
- Regla de definición de claves.
- Regla de agregación de rectángulos.
- Regla de supresión de rectángulos.
- Representación de asociaciones n-arias.
- Asociaciones explícitas parciales.
- Producto de asociaciones.
- Dependencias plurales.
- Otros tipos de dependencias.

En el capítulo siguiente se resume el método de diseño propuesto y se presentan algunos ejemplos.

REPRESENTACION DE CONJUNTOS DE VALORES

Empezaremos representango gráficamente conjuntos de valores simples, es decir, conjuntos de símbolos que tienen un cierto significado, definido porque cumplen una cierta propiedad o predicado. Este conjunto es una relación unaria, R(X), donde el nombre de la relación, R, es normalmente una abreviatura de la propiedad que define el significado de los valores X.

La representación gráfica del conjunto R(X) será un rectángulo con la etiqueta R(X) en su interior, o a veces, si no nos interesa indicar explícitamente el nombre de la propiedad R, con sólo el nombre del atributo X en su interior:



A veces interesa considerar conjuntos de valores que son subconjuntos de otros. Esto lo indicaremos dando el mismo nombre al atributo de las relaciones unarias correspondientes. No es imprescindible hacerlo así para aplicar el proceso de diseño que vamos a describir a lo largo de este capítulo, pero ayuda a documentarlo y hacerlo más claro. Así, si partimos de R(X) y extraemos del conjunto R un subconjunto de símbolos X que cumplan un cierto predicado, S, tendremos una relación unaria que representaremos S(X).

Ejemplo:

Sea E(SS) el conjunto de los números de Seguridad Social de todos los empleados de una empresa. El predicado E significa «empleado». Los valores del atributo, SS, son símbolos formados por dígitos que representan números de afiliados a la Seguridad Social. Consideremos ahora un subconjunto de E formado por todos los números de Seguridad Social de los empleados que están casados. Lo representaremos como C(SS) (predicado C: casados; atributo SS: números de Seguridad Social).

Si consideramos el conjunto de todos los DNI (Documento Nacional de Identidad) de los empleados casados tendríamos una relación unaria diferente, en la que hay que dar al atributo un nombre distinto de SS, puesto que es un conjunto de símbolos que no está contenido en E(SS). En cambio, sería aconsejable mantener el mismo nombre del predicado, puesto que tiene el mismo significado (C: casados), aunque no es imprescindible. Así, esta nueva relación podríamos llamarla C(DNI).

Consideremos ahora el conjunto de los números de Seguridad Social de los proveedores de la empresa. Aunque no es un subconjunto de E(SS), conviene dar al atributo el mismo nombre, SS, pues ambos conjuntos pueden considerarse subconjuntos de otro más amplio (el conjunto de los números de Seguridad Social de todos los ciudadanos del país). De forma que lo podríamos llamar PROV(SS) (PROV: proveedores; SS: número de Seguridad Social).

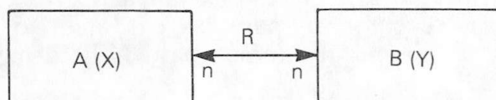
Estos convenios de nombres son útiles para documentar y clarificar el proceso de diseño, como ya se ha dicho, pero no es imprescindible atenerse a ellos.

Entre los elementos de estos conjuntos pueden existir asociaciones que expresan ciertas propiedades o predicados. El grado de una asociación es el número de conjuntos que intervienen en ella (asociaciones binarias, ternarias, etcétera).

REPRESENTACION DE ASOCIACIONES BINARIAS

Representaremos a una asociación binaria entre dos conjuntos de valores mediante una línea que una a los rectángulos participantes. En cada extremo de esta línea pondremos un símbolo, de acuerdo con una notación que veremos a

continuación, para indicar la «cardinalidad» de la asociación en ambos sentidos (o sea, el número de elementos asociados a uno dado). Si en alguno de los extremos es la cardinalidad mayor que 1 se dibujará en él una flecha. Además, a cada asociación se le da un nombre que suele ser el predicado o una abreviatura de él. Se escribirá este nombre al lado de la línea. Ejemplo:



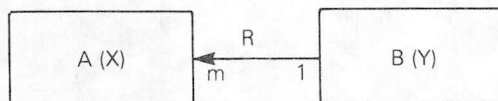
Significado de los elementos de esta figura:

- A(X): Conjunto de valores que cumplen la propiedad A.
- B(Y): Conjunto de valores que cumplen la propiedad B.
- R: Nombre del predicado que cumplen las parejas de valores X e Y asociados.
- n: Cardinalidad.

Para expresar la cardinalidad de la asociación emplearemos la notación siguiente:

- n: Indica que a un elemento de un conjunto le pueden corresponder 0, 1, 2, ..., n, elementos del otro (es una asociación múltiple que incluye el cero).
- m: Indica que a un elemento le pueden corresponde 1, 2, ..., n (asociación múltiple excluyendo el cero).
- 1: Indica que a cada elemento le corresponde otro.
- c: Indica que a un elemento le pueden corresponder 0 ó 1 («c» viene de «condicional»).

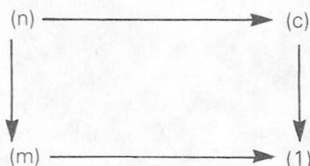
Cuando queremos representar la cardinalidad de una asociación, lo hacemos incluyendo entre paréntesis, y separados por dos puntos, dos de los símbolos anteriores. El de la izquierda representará la cardinalidad yendo de derecha a izquierda, y el otro en el sentido inverso. Así, por ejemplo, si decimos que la asociación entre A(X) y B(Y) es de cardinalidad (m : 1), estamos indicando que a todo elemento Y de B le corresponde un número de elementos X de A que puede ser 1, 2, ..., o n, y que a cada elemento X de A le corresponde exactamente otro de B. La representación gráfica sería:





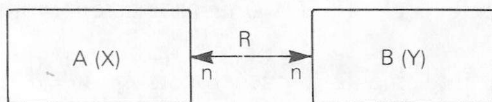
Con estos símbolos podremos tener diez tipos posibles de asociaciones binarias: (n : n), (n : m), (n : c), (n : 1), (m : m), (m : c), (m : 1), (c : c), (c : 1), (1 : 1).

Puesto que el conjunto de valores incluidos en la cardinalidad representada con n incluye a los representados con m y c, y éstos a su vez al valor representado con la cardinalidad 1, convendremos en que, de ahora en adelante, y salvo que se diga expresamente lo contrario, cuando enunciemos una propiedad para la cardinalidad n también se cumplirá para las cardinalidades m, c y 1, y las propiedades de m y c se cumplirán también para 1. Es decir, si la flecha se lee «implica»:



Evidentemente, una asociación binaria R entre dos conjuntos A(X) y B(Y), podremos reflejarla en el esquema relacional de diseño mediante una relación binaria en la que cada tupla está formada por las parejas de valores ⟨X, Y⟩ asociados. En esta relación binaria usaremos los mismos nombres de atributos, X e Y, que en las relaciones participantes.

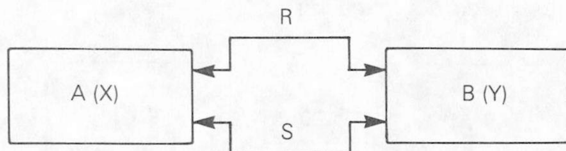
Por tanto, el diagrama



dará lugar al siguiente esquema relacional de diseño:

- A(X) (conjunto de valores X que cumplen el predicado A).
- B(Y) (conjunto de valores Y que cumplen el predicado B).
- R(X, Y) (conjunto de parejas de valores de X e Y que cumplen el predicado R).

Es posible que exista más de una asociación entre los mismos rectángulos:



En este caso el esquema de diseño sería:

A(X), B(Y), R(X, Y), S(X, Y)

Ejemplo 1:

Consideremos un centro de enseñanza que asigna un número de matrícula, MAT, a sus alumnos. Tendríamos los conjuntos de valores siguientes:

P(DNI): Conjunto de los DNI's de los profesores del centro.

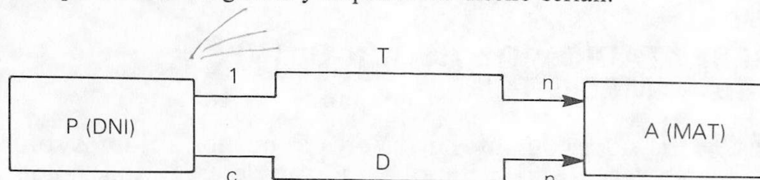
A(MAT): Conjunto de los números de matrícula de los alumnos del centro.

Podríamos considerar las dos asociaciones siguientes:

T(DNI, MAT): Profesores que son tutores de alumnos.

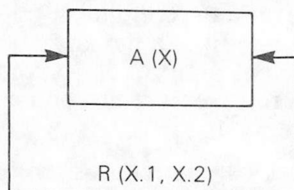
D(DNI, MAT): Profesores que dirigen tesis doctorales a alumnos.

Los correspondientes diagrama y esquema de diseño serían:



P(DNI), A(MAT), T(DNI, MAT), D(DNI, MAT)

También puede ocurrir que las parejas de valores que se asocian se tomen de un mismo conjunto, A(X). El diagrama sería en este caso:

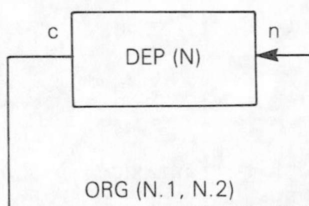


En el diagrama anterior hemos calificado al atributo X para distinguir los dos papeles que desempeña en R. El esquema de diseño sería:

A(X), R(X.1, X.2)

Ejemplo 2:

Consideremos el conjunto de valores, DEP(N), de todos los números de departamento de una empresa. Entre ellos existe una asociación definida por la dependencia organizativa de unos respecto a otros. Así, la asociación ORG (N.1, N.2) indica qué departamentos (N.1) dependen de otros (N.2) en el organigrama de la empresa. Los correspondientes diagrama y esquema de diseño serían:



DEP(N), ORG(N.1, N.2)

Ejemplo 3:

Sea E (DNI) el conjunto de DNI's de los empleados de una empresa. Consideremos el predicado «Jefe» que nos asocia a cada jefe con sus subordinados. Puede representarse esta asociación con la relación JEFE (DNI.J, DNI.S). El diagrama y el esquema de diseño serían análogos a los ya vistos anteriormente.

REPRESENTACION DE ASOCIACIONES BINARIAS IMPLICITAS

Consideremos dos conjuntos de valores en que los atributos son comparables, es decir, homogéneos. El criterio para considerar si son homogéneos o no puede ser tan amplio como se desee. Por ejemplo, podríamos considerar que son homogéneos si ambos son alfabéticos o si ambos son numéricos, o si son cantidades expresadas en la misma unidad (por ejemplo, pesetas), o si toman valores en el mismo dominio, es decir, si significan lo mismo (por ejemplo, ambos son números de DNI), etc.

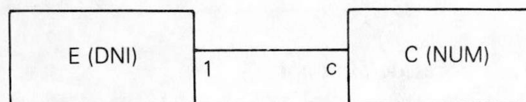
Una vez definido un criterio de homogeneidad o comparación, podemos definir una asociación entre ambos conjuntos que ligue los valores que son iguales. A toda asociación de este tipo la llamaremos «implícita», y la representaremos mediante una línea sin nombre de predicado, entre los rectángulos asociados. La cardinalidad la representaremos con el mismo convenio ya descrito anteriormente.

Normalmente sólo estaremos interesados en representar las asociaciones implícitas cuando los atributos tengan el mismo significado. Convendremos, de ahora en adelante, que en este caso daremos el mismo nombre a los atributos para hacer más evidente la existencia de la asociación implícita.

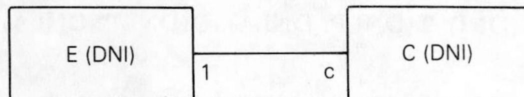
Ejemplo 1:

Supongamos que tenemos los conjuntos:
E(DNI): Conjunto de números de DNI de los empleados.
C(NUM): Conjunto de números de DNI de los empleados casados.

En este ejemplo suponemos que no estamos siguiendo el convenio de dar el mismo nombre a atributos con igual significado, pues DNI y NUM son ambos números de DNI. Existe entre ellos una asociación implícita que representaremos así:



Si seguimos el convenio de nombres tendremos:



Evidentemente, la cardinalidad de una asociación implícita no puede ser múltiple. Es decir, sólo puede ser c, 1 ó 0. Una cardinalidad 0 significa que no hay valores comunes a ambos conjuntos (su intersección es vacía) y, por tanto, podemos omitir la línea entre los rectángulos (o poner como cardinalidad (0:0) si queremos indicar esta condición explícitamente).

El objetivo de representar las asociaciones implícitas es poder indicar su cardinalidad en el dibujo.

Ejemplo 2:

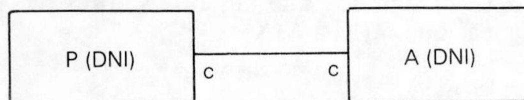
Supongamos que tenemos los conjuntos siguientes en un centro de enseñanza:

P(DNI): Conjunto de DNI's de los Profesores.

A(DNI): Conjunto de DNI's de los Alumnos.

Si suponemos que ningún alumno puede ser profesor, no habrá valores comunes entre los conjuntos P y A (cardinalidad cero).

En caso contrario, sí los habrá. En nuestro ejemplo podríamos tener:

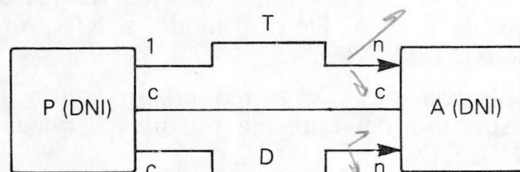


Esta cardinalidad indica que algunos profesores pueden ser alumnos, y viceversa.

Naturalmente, entre dos conjuntos con el mismo atributo, puede haber, además de la asociación implícita, otras explícitas.

Ejemplo 3:

Volvamos a nuestro ejemplo anterior del centro de enseñanza. Podríamos tener el diagrama:



donde:

T: Profesores tutores de alumnos.

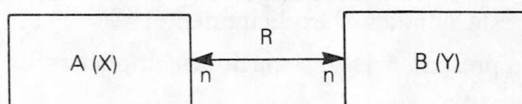
D: Profesores que dirigen tesis doctorales de alumnos.

El esquema de diseño relacional sería:

P(DNI), A(DNI), T(DNI.P, DNI.A), D(DNI.P, DNI.A)

CARDINALIDAD Y CONDICIONES DE INTEGRIDAD

Como ya hemos visto, cuando tenemos una asociación binaria explícita, tal como:



el esquema de diseño está formado por tres relaciones:

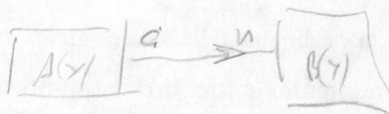
A(X), B(Y), R(X, Y)

Observemos, sin embargo, que en este esquema no se ha reflejado para nada la cardinalidad de R, dato que sí consta en el diagrama anterior. Para reflejar esto hay que introducir condiciones de integridad en el esquema. Veamos cómo:

1) Si la asociación R tiene en un extremo, por ejemplo, en el B(Y), la cardinalidad n (o c) significa que algunos valores de X en A pueden no estar en R. O sea, que **R**[X]⊆A(X).

Ejemplo de valores:

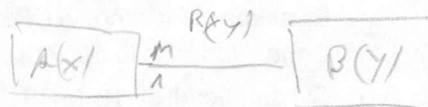
A(X)	B(Y)	R(X, Y)
a	1	(a, 1)
b	2	(a, 2)
c	3	(c, 3)
d	4	(c, 4)
e	5	(e, 5)



2) Si la asociación R tiene en un extremo, por ejemplo en el B(Y), la cardinalidad m (ó 1) significa que todos los valores de X en R son los mismos que los de X en A. De otro modo: si R(X, Y) es (n : m) o (n : 1), se verifica que **R**[X] = A(X).

Por tanto, la relación A(X) es redundante con la R(X, Y) y se puede en principio suprimir del esquema de diseño, quedando éste con dos relaciones: B(Y), R(X, Y).

Ejemplo de valores:



A(X)	B(Y)	R(X, Y)
a	1	(a, 1)
b	2	(b, 1)
c	3	(b, 3)
d	4	(c, 4)
	5	(d, 5)
	6	(d, 6)

3) Si la asociación R tiene en un extremo, por ejemplo en el B(Y), la cardinalidad c (ó 1) significa que existe una DF (X→Y) en R. En efecto, como a cada valor de X en R sólo le corresponde un Y, los valores de X no pueden repetirse en R, es decir, X es la clave de R, y por tanto además X→Y.

Ejemplo de valores:

A(X)	B(Y)	R(X, Y)
a	1	(a, 1)
b	2	(b, 1)
c	3	(c, 3)
d	4	
	5	

Aplicando estas ideas tendremos los siguientes esquemas de diseño para los diez tipos posibles de cardinalidad:

1) Cardinalidad de R: (n : n).
Esquema de diseño: A(X), B(Y), R(X, Y).
Cond. de integridad: **R**[X]⊆A(X); **R**(Y)⊆B(Y).

2) Cardinalidad de R: (n : m).
Esquema de diseño: B(Y), R(X, Y).
Cond. de integridad: **R**[Y]⊆B(Y).

3) Cardinalidad de R: (n : c).
Esquema de diseño: A(X), B(Y), R(X, Y).
Cond. de integridad: **R**[X]⊆A(X); **R**[Y]⊆B(Y); Clave de R: X.

4) Cardinalidad de R: (n : 1).
Esquema de diseño: B(Y), R(X, Y).
Cond. de integridad: **R**[Y]⊆B(Y); Clave de R: X.

5) Cardinalidad de R: (m : m).
Esquema de diseño: R(X, Y).
Cond. de integridad: Ninguna.

6) Cardinalidad de R: (m : c).
Esquema de diseño: A(X), R(X, Y).
Cond. de integridad: **R**[X] ⊆ A(X); Clave de R: X.

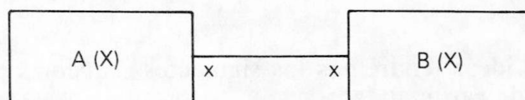
7) Cardinalidad de R: (m : 1).
Esquema de diseño: R(X, Y).
Cond. de integridad: Clave de R: X.

8) Cardinalidad de R: (c : c).
Esquema de diseño: A(X), B(Y), R(X, Y).
Cond. de integridad: **R**[X] ⊆ A(X); **R**[Y] ⊆ B(Y); Claves de R: X e Y.

9) Cardinalidad de R: (c : 1).
Esquema de diseño: B(Y), R(X, Y).
Cond. de integridad: **R**[Y] ⊆ B(Y); Claves de R: X e Y.

10) Cardinalidad de R: (1 : 1).
Esquema de diseño: R(X, Y).
Cond. de integridad: Claves de R: X e Y.

Todo esto es también aplicable a las asociaciones implícitas. Tendríamos los casos siguientes:



1) Cardinalidad: (c : c).
Esquema de diseño: A(X), B(X).
Cond. de integridad: Ninguna.

2) Cardinalidad: (c : 1).
Esquema de diseño: A(X), B(X).
Cond. de integridad: A(X) ⊆ B(X).

3) Cardinalidad: (1 : 1).
Esquema de diseño: A(X).
Cond. de integridad: A(X) = B(X).

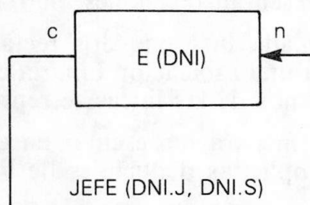
4) Cardinalidad: (0 : 0).
Esquema de diseño: A(X), B(X).
Cond. de integridad: No puede haber valores comunes entre A(X) y B(X).

Con las reglas expuestas podemos obtener el esquema relacional de diseño correspondiente a una asociación binaria incluyendo las condiciones de integridad deducibles de la cardinalidad. Esto no significa que no pueda haber otras

condiciones de integridad, no expresables mediante la cardinalidad de la asociación.

Ejemplo

Reconsideremos la asociación JEFE entre los empleados de una empresa. El diagrama es:



El esquema relacional de diseño será:

Relaciones: E (DNI), JEFE (DNI.J, DNI.S).

Condiciones de integridad: Clave de JEFE: (DNI.S).

No obstante, hay que cumplir una condición adicional, no expresada por la cardinalidad (n : c), que es que sólo un empleado no tiene jefe (el jefe supremo o jefe de todos los jefes). Sean, por ejemplo, los valores siguientes:

E (DNI)	JEFE (DNI.J, DNI.S)
1	(1, 2)
2	(1, 3)
3	(2, 4)
4	(2, 5)
5	(5, 6)
6	(5, 7)
7	

Sólo el empleado 1 no figura como subordinado en la relación JEFE.

Como hemos visto, las asociaciones binarias en que la cardinalidad en un extremo no supera a 1 dan lugar a una Dependencia Funcional y, consiguientemente, a una clave. Por ello las llamaremos «asociaciones funcionales». Clasificaremos entonces a las asociaciones binarias, según su cardinalidad, en tres tipos:

1) Asociaciones plurales: (n : n), (n : m), (m : m).

2) Asociaciones funcionales:

a) Asociaciones funcionales completas: (n : 1), (m : 1), (c : 1), (1 : 1).

b) Asociaciones funcionales incompletas: (n : c), (m : c), (c : c).

5

DIAGRAMA DE CONJUNTOS Y ASOCIACIONES

Con los elementos analizados hasta aquí podemos representar gráficamente conjuntos de valores y asociaciones binarias entre ellos. Al diagrama así formado lo llamaremos diagrama C/A (diagrama de Conjuntos y Asociaciones). Un diagrama C/A está formado por tanto por los elementos siguientes:

1) *Rectángulos*, que representan relaciones unarias o conjuntos de valores.
2) *Líneas con nombre*. Cada una une dos rectángulos (no necesariamente distintos) y representa una asociación binaria explícita entre los elementos de éstos. En los extremos de las líneas se representa la cardinalidad.
3) *Líneas anónimas*. Sirven para representar en el diagrama la cardinalidad de las asociaciones implícitas deducibles de los atributos comunes entre los conjuntos.

A partir de un diagrama C/A se puede obtener el esquema relacional de diseño de la siguiente forma:

1) Cada rectángulo da lugar a una relación unaria.
2) Cada línea con nombre da lugar a una relación binaria.
3) Las cardinalidades de las líneas dan lugar a condiciones de integridad (claves e integridad de referencia).

La representación de los datos mediante el diagrama C/A, tal como la hemos descrito hasta aquí, presenta algunos inconvenientes y limitaciones:

1) Es demasiado prolija. Conviene dotarla de mayor concisión.
2) No permite representar asociaciones binarias en las que participen elementos de otras asociaciones.
3) No permite representar asociaciones de grado superior a 2.

Vamos a introducir algunas modificaciones para resolver estos inconvenientes. Para ello tomaremos como elementos básicos de construcción los rectángulos, para representar relaciones n-arias, y las asociaciones funcionales completas. Estas nos van a permitir definir las entidades y representar asociaciones de orden superior.

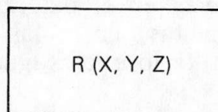
Estas modificaciones serán el resultado de aplicar unas reglas de transformación, que son las siguientes:

1) Propagación de claves.
2) Definición de claves.
3) Agregación de rectángulos.
4) Supresión de rectángulos.

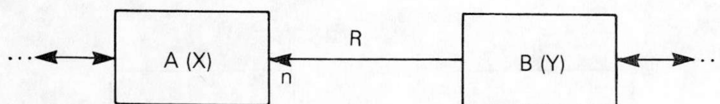
5

REGLA DE PROPAGACION DE CLAVES

Hasta ahora hemos usado los rectángulos en los diagramas para representar conjuntos de valores simples, es decir, relaciones unarias. A partir de ahora vamos a utilizarlos para representar, en general, conjuntos de valores simples o compuestos, es decir, relaciones de grado cualquiera. Así, por ejemplo, la relación R(X, Y, Z) la podemos representar así:



Este rectángulo representa el conjunto de tuplas de la relación R. Supongamos entonces que entre dos rectángulos, A(X) y B(Y), existe una asociación funcional completa explícita, R(X, Y):

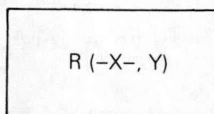


(Diagrama (1)

En el diagrama anterior podría haber, en general, otras asociaciones binarias entre A y B, además de R. Estas asociaciones podrían ser de A y B con otros o incluso entre sí. Esto se indica en la figura anterior con puntos. El esquema relacional de diseño será:

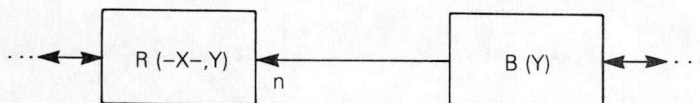
R(Y), R(-X-, Y); Condiciones: **R**[Y]⊆B(Y). (Nota: en la relación R, X es la clave. Esto lo representamos poniendo la X entre guiones: R(-X-, Y)).

Por lo dicho anteriormente, el rectángulo:



representa el conjunto de tuplas de la relación R(X, Y), indicando además cuál es su clave primaria.

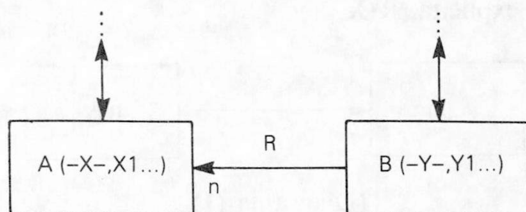
Entonces podemos substituir el rectángulo A(X) del diagrama (1) anterior por el rectángulo R, quedando este diagrama transformado así:



Además de reemplazar el rectángulo A por el R, hemos transformado la línea R que unía a A con B en una línea anónima, y hemos mantenido todas las demás líneas que confluían en A tal como estaban. Es importante subrayar que hemos transformado una *ASOCIACION EXPLICITA* R en una asociación *IMPLICITA*, es decir, que si aplicamos esta transformación a todas las asociaciones explícitas del diagrama, al final sólo habrá en éste asociaciones binarias, funcionales completas e implícitas.

Todas las líneas que confluyen en el diagrama anterior en R representan asociaciones (explícitas o implícitas) entre las tuplas de R y las de otros rectángulos (o las suyas propias) o, lo que es igual, entre los valores de la clave de R y otras claves.

Sean, en general, dos rectángulos, A(X, X1, ...) y B(Y, Y1, ...), y una asociación funcional completa R entre sus tuplas o, lo que es lo mismo, entre sus claves:



El esquema relacional de diseño será:

Relaciones: A(-X-, X1, ...), B(-Y-, Y1, ...), R(-X-, Y).

Condiciones: **R**[X] = **A**[X], **R**[Y] ⊆ **B**[Y].

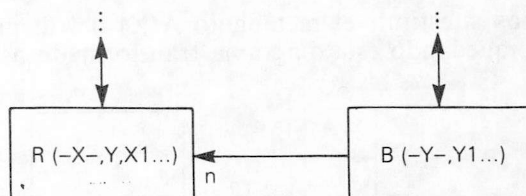
En definitiva, llamaremos «propagar» la clave Y del rectángulo B, al rectángulo A, a la transformación siguiente:

1) El rectángulo A lo reemplazamos por R(-X-, Y, X1, X2, ...), donde esta relación es el resultado de (A(-X-, X1, ...) * R(-X-, Y)).

2) Todas las líneas que confluyen en A confluyen ahora en el rectángulo R que lo substituye.

3) La asociación R explícita que unía a los rectángulos A y B, se transforma en implícita.

El diagrama quedaría transformado en el siguiente:



En este dibujo la línea anónima entre los rectángulos R y B indica que, si asociamos cada tupla de R con todas las de B en que el atributo común, Y, tiene igual valor, cada tupla de R resultará asociada a una, y sólo una, de B, y cada tupla de B resultará asociada a ninguna, una o varias, de R.

El esquema relacional de diseño será:

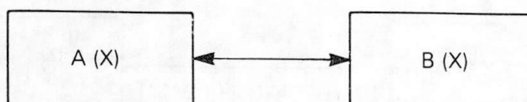
Relaciones: B(-Y-, Y1, ...), R(-X-, Y, X1, ...).
Condiciones: Integridad de referencia: Todo valor de Y en R debe existir en B.

ASOCIACIONES IMPLICITAS ENTRE RELACIONES UNARIAS

Ya hemos visto que entre dos relaciones que tengan atributos comunes existe una asociación binaria implícita. Vamos a recapitular lo dicho sobre este tipo de asociaciones y a considerar algunas propiedades adicionales.

Sean dos relaciones unarias, A(X) y B(X), con el mismo atributo, X. La asociación implícita entre ellas es la que se obtiene emparejando los valores de X que son iguales en A y en B.

Puesto que la existencia de este tipo de asociaciones es directamente deducible a partir de la existencia de un atributo común, es en principio redundante representarlas en el diagrama. Sin embargo, es útil hacerlo para poder indicar sobre ellas la cardinalidad de la asociación. Para representar una asociación de este tipo dibujamos una línea «anónima».



Evidentemente la cardinalidad no puede superar a 1. Por tanto, los casos posibles de cardinalidad son:

(0 : 0): A(X) y B(X) son conjuntos disjuntos.

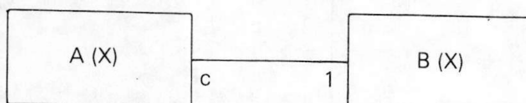
(1 : 1): Ambos conjuntos son iguales, A(X) = B(X).

(c : 1): Todos los valores de A(X) están en B(X), pero no al revés necesariamente. Es decir, A(X) ⊆ B(X).

(c : c): Puede haber valores comunes, o no, entre A y B.

Ejemplo:

El diagrama:



dará lugar al esquema de diseño:

Relaciones: A(X), B(X).

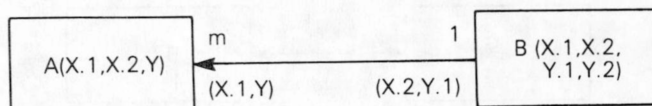
Condiciones: Todos los valores de X en A deben estar en B.

ASOCIACIONES IMPLICITAS ENTRE RELACIONES N-ARIAS

Sean ahora dos relaciones n-arias, A(X, Y, Z) y B(X, Y, W), donde X, Y, Z y W son atributos posiblemente compuestos. Por el hecho de tener atributos comunes existen asociaciones implícitas entre ambas, que se obtienen emparejando las tuplas de A y B que tienen iguales valores en todos, o en parte, de los atributos comunes. Si en todos, diremos que se trata de una asociación implícita total, y si no, parcial. Normalmente, mientras no se diga lo contrario, al hablar de una asociación implícita supondremos que es total.

Las asociaciones implícitas las representaremos por una línea anónima sobre cuyos extremos se indicarán las cardinalidades respectivas, así como los atributos que intervienen en ella. Estos últimos pueden omitirse si la asociación implícita es total y no hay ambigüedad para designarlos. La cardinalidad puede venir dada, en general, por una pareja formada por cualquiera de las diez combinaciones posibles entre los valores n, m, c y 1, o la pareja (0:0) si no hay valores comunes.

Consideremos el siguiente diagrama:



Este diagrama representa que la asociación implícita que empareja las tuplas de A y B cuando coinciden los valores de (X.1, Y) en A con los de (X.2, Y.1) en B, tiene una cardinalidad (m:1).

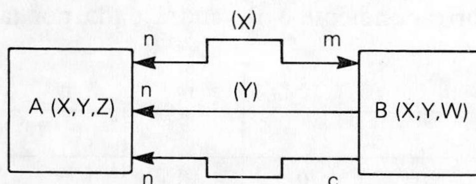
Ejemplo 1:

Sean dos relaciones A(X, Y, Z) y B(X, Y, W). Los siguientes valores corresponderían a una asociación de tipo (n:c) según los valores de (X, Y), de tipo (n:m) según los valores de (X) y de tipo (n:1) según los valores de (Y):

A	X	Y	Z
	1	a	p
	2	a	q
	2	b	r
	3	d	s
	3	d	t

B	X	Y	W
	1	a	10
	1	b	11
	2	c	12
	3	d	13
	4	e	14
	4	e	15

El diagrama correspondiente sería:

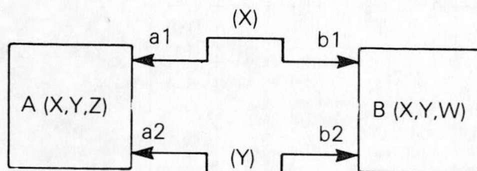


Normalmente sólo estaremos interesados en representar la asociación implícita total.

Obsérvese que una asociación implícita empareja las mismas tuplas de A y B que se emparejan también al ayuntar ambas relaciones por sus atributos comunes. Por ello la cardinalidad de la asociación nos da una primera aproximación del coste de la yunción. Si, por ejemplo, la asociación implícita entre A(X, Y) y B(X, Z) es (n : 1), el número de tuplas de (A * B) es igual al de A. Si fuera (n : c), el número de tuplas de (A * B) no puede superar al de A. Si fuera (n : n), y la cardinalidad media fuera, por ejemplo, (3 : 2) (o sea, a cada tupla de A corresponden, en promedio, dos de B, y a cada una de B, 3 de A), cada valor de X común a A y B estará repetido, en promedio, en 6 tuplas de (A * B).

Los valores de la cardinalidad de la asociación implícita total están condicionados por los de las asociaciones implícitas parciales. Veamos esto.

Sea el diagrama:



Tenemos, pues, que la cardinalidad según (X) es (a1 : b1), y según (Y) es (a2 : b2). Vamos a ver qué valores puede tener la cardinalidad según (X, Y).

A una tupla de A le corresponden, en B, b1 tuplas con igual valor en X, y b2 tuplas con igual valor en Y, pero es posible que no coincidan en ninguna de estas tuplas X e Y a la vez, o es posible que sí. El número de tuplas en las que sí pueden coincidir viene dado, en función de b1 y b2, por la tabla siguiente:

b2:	0	1	c	m	n
b1:					
0	0	0	0	0	0
1	0	0 ó 1	0 ó 1	0 ó 1	0 ó 1
c	0	0 ó 1	0 ó 1	0 ó 1	0 ó 1
m	0	0 ó 1	0 ó 1	0-n	0-n
n	0	0 ó 1	0 ó 1	0-n	0-n

Por tanto, los valores posibles para la cardinalidad de la asociación implícita total en el extremo correspondiente a B vendrá dada por la tabla siguiente:

b2:	0	1	c	m	n
b1:					
0	0	0	0	0	0
1	0	c, 1, 0	c, 0	c, 1, 0	c, 0
c	0	c, 0	c, 0	c, 0	c, 0
m	0	c, 1, 0	c, 0	n, m, c, 1, 0	n, c, 0
n	0	c, 0	c, 0	n, c, 0	n, c, 0

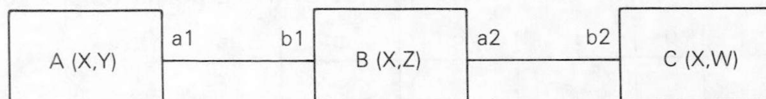
Ejemplo 2:

Veamos un caso en que la asociación parcial según X es (m : 1), según Y es (m : c) y según (X, Y) es (n : c):

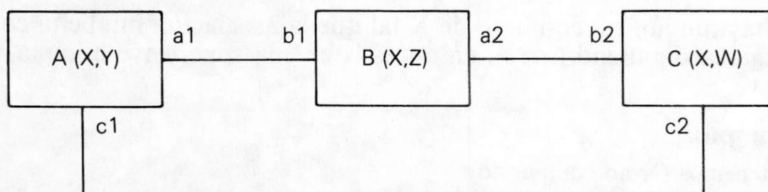
A	X	Y	Z
	1	a	p
	1	a	q
	2	b	q
	2	c	q
	2	d	r
	3	a	r

B	X	Y	W
	1	a	10
	2	b	11
	3	d	12

Otro caso interesante que se presenta en los diagramas es el de asociaciones implícitas compuestas a partir de otras, cuya cardinalidad también está condicionada por éstas. Precisemos esto. Sea el diagrama:



En este diagrama la cardinalidad de la asociación implícita entre A y B es (a1 : b1), y entre B y C es (a2 : b2). Obviamente, puesto que X es un atributo común a A y B, y a B y C, también lo es a A y C, por lo que podemos dibujar una asociación implícita entre éstos, y diremos que esta asociación es el resultado de componer las asociaciones A-B y B-C. Sea (c1 : c2) la cardinalidad de la asociación compuesta:

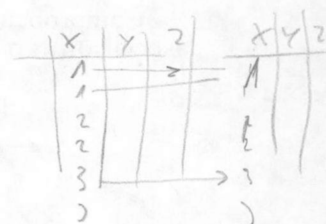


Si b1 es 1 ó m, es decir, si todos los valores de X en A existen en B, los valores posibles de c1 vienen dados por la tabla siguiente:

a2: / a1:	0	1 ó m	c ó n
1	0	1	c
c	0	c, 1, 0	c, 0
m	0	m, 1	n, c
n	0	n, m, c, 1, 0	n, c, 0

Si b1 es c ó n, es decir, si algunos valores de X en A no existen en B, los valores posibles de c1 vienen dados por la tabla siguiente:

a2: / a1:	0	1 ó m	c ó n
1	n, m, c, 1, 0	1	n, m, c, 1
c	n, m, c, 1, 0	c, 1, 0	n, m, c, 1, 0
m	n, m, c, 1, 0	m, 1	n, m, c, 1
n	n, m, c, 1, 0	n, m, c, 1, 0	n, m, c, 1, 0



REGLA DE DEFINICION DE CLAVES

Otra propiedad interesante de las asociaciones implícitas es que la cardinalidad (m : c), ó (m : 1), ó (1 : c), ó (1 : 1) define la existencia de una superclave.

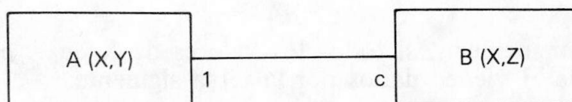
Sean dos relaciones n-arias, A(X, Y, Z) y B(X, Y, W), con atributos comunes X e Y (posiblemente compuestos). Si la asociación implícita según X entre A y B es (m : c), X es una superclave en B.

En efecto, si no lo fuera, podría tener X valores repetidos en B, y habría por consiguiente más de una tupla de B asociada a una de A, lo que contradice a la cardinalidad c.

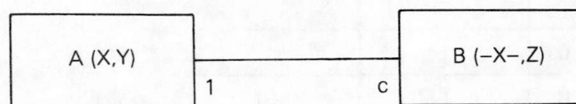
Si no hay ningún subconjunto de X tal que la asociación implícita correspondiente tenga cardinalidad (m : c), entonces X es una superclave mínima, es decir, es una clave.

Ejemplo:

Si tenemos (X no compuesto):



X debe ser clave en B, es decir:

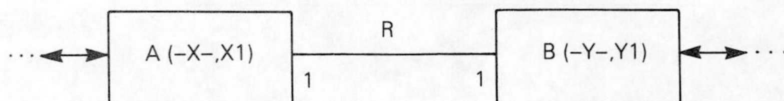


El recíproco no es cierto. Sin embargo, sí puede afirmarse que si X es una superclave de A, e interviene en una asociación implícita, la cardinalidad de ésta no puede ser múltiple (es decir, m o n). Así, si tenemos A(-X-, Y) y B(X, Z), la asociación implícita entre A y B ha de tener cardinalidad c ó 1 en el extremo A.

REGLA DE AGREGACION DE RECTANGULOS

En el caso de una asociación funcional completa biunívoca, explícita o implícita, podemos fundir los dos rectángulos en uno. A esta transformación del diagrama la llamaremos «agregar» dos rectángulos.

Comprobemos la justificación de esta regla. Consideremos primero una asociación explícita:



El esquema relacional de diseño será:

Relaciones: A(-X-, X1), B(-Y-, Y1), R(-X-, -Y-).

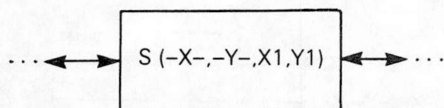
Condiciones: **A**[X] = **R**[X]; **B**[Y] = **R**[Y].

Este diseño es equivalente a:

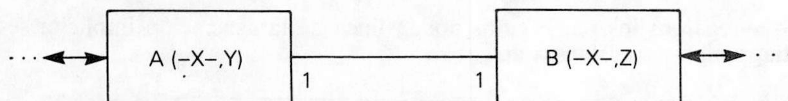
Relaciones: S(-X-, -Y-, X1, Y1), donde S = A * R * B.

Condiciones: ∅.

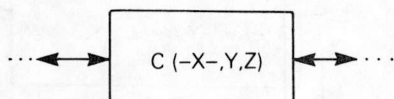
Por tanto, el diagrama puede transformarse substituyendo A y B por S:



Consideremos ahora una asociación implícita:



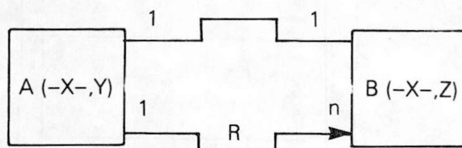
Ambos rectángulos se pueden subtituir por C = A * B:



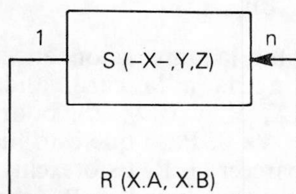
Al agregar dos rectángulos, todas las líneas (asociaciones) que confluyen en los agregandos, confluirán en el rectángulo resultante y se conservarán en éste las claves originales.

Ejemplo 1:

Sea un diagrama con una asociación implícita (1 : 1) y otra explícita (1 : n):



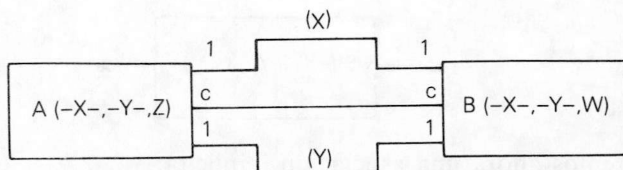
Agregando los rectángulos se transforma en:



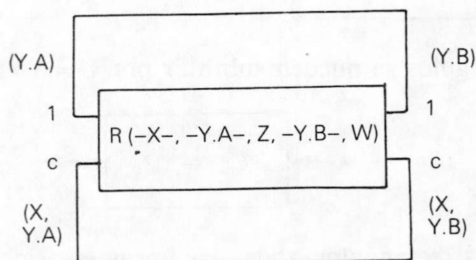
(donde S = A * B).

También puede ocurrir que la asociación implícita usada para la agregación sea una asociación parcial.

Ejemplo 2:

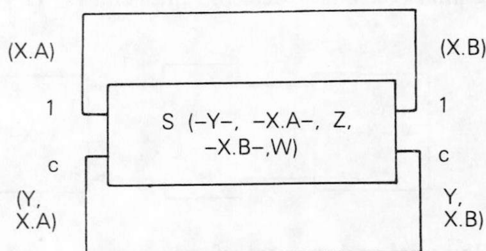


Si agregamos los rectángulos por la línea de la asociación implícita según (X) el diagrama se transforma en:



(donde R = AYB (X.A = X.B)).

Si ahora agregamos los rectángulos por la línea de la asociación implícita según (Y) el diagrama se transforma en:



(donde S = AYB(Y.A = Y.B)).

Podría pensarse en intentar la agregación en el caso de una cardinalidad cualquiera, en vez de restringirla a la cardinalidad (1 : 1). Si tenemos, por ejemplo, dos rectángulos A(X, Y) y B(X, Z), podríamos intentar substituirlos por el R(X, Y, Z), siendo R = A * B. Para que esto fuera válido, todos los valores de X en A o en B deberían aparecer en R. Esto excluye las cardinalidades n y c. Es decir, que la asociación implícita entre A y B debe ser (m : m) o (m : 1). Pero esto nos llevaría a relaciones no normalizadas. En efecto, como R, debe ser es descomponible reversiblemente en A y B, se verifica en R que $X \twoheadrightarrow Y \mid Z$, y como X no es clave en R, pues si lo fuera tendría que ser (1 : 1) la cardinalidad de la asociación entre A y B, se deduce que R no está normalizada. Por tanto, no haremos este tipo de transformaciones.

En resumen, la regla de agregación de rectángulos se aplicará a asociaciones de cardinalidad (1 : 1), explícitas o implícitas.

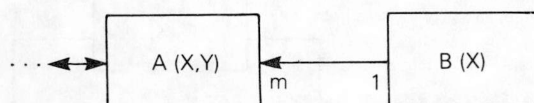
REGLA DE SUPRESION DE RECTANGULOS

Si un rectángulo participa con todos sus atributos en una o más asociaciones implícitas, de las que una al menos es de cardinalidad (1 : m), puede suprimirse del diagrama.

Justifiquemos esto con algunos ejemplos:

Ejemplo 1:

Sea el diagrama (X es posiblemente compuesto):

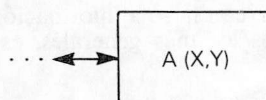


El esquema de diseño relacional es:

Relaciones: A(X, Y), B(X).

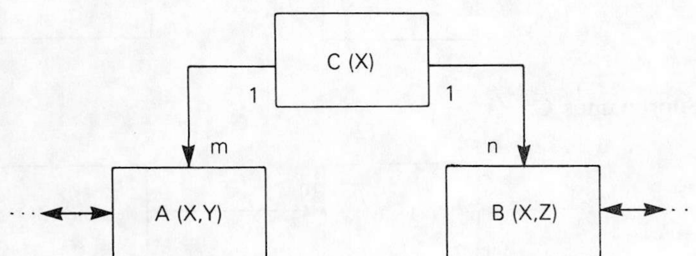
Condiciones: A[X] = B[X].

Es decir, que B(X) es redundante, por lo que puede suprimirse. Esto se refleja en el diagrama transformándolo en el siguiente:



Ejemplo 2:

Veamos el caso en que el rectángulo que se suprime participa en más de una asociación. Sea el diagrama:

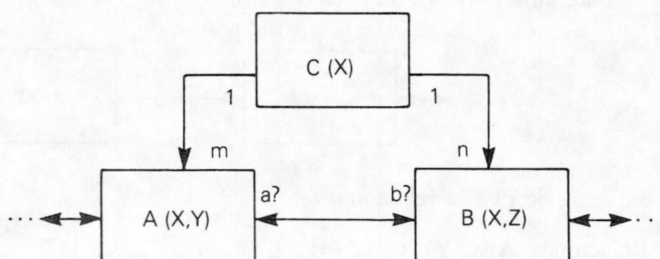


En este diagrama, como siempre, X, Y y Z pueden ser atributos compuestos. El rectángulo C participa en dos asociaciones, con A y con B, en ambas con todos sus atributos, y una de ellas es de cardinalidad (1 : m). Por tanto, según la regla anterior, es suprimible. En efecto, el esquema de diseño es:

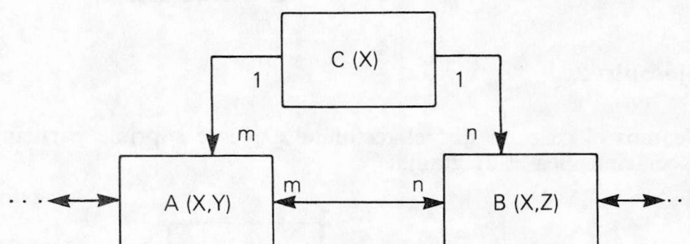
Relaciones: A(X, Y), B(X, Z), C(X).

Condiciones: $\mathbf{A}[X] = C(X)$; $\mathbf{B}[X] \subseteq C(X)$.

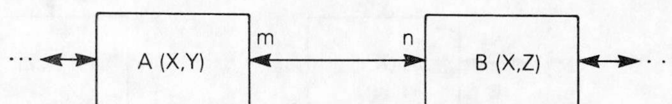
Por tanto, C(X) es redundante con A(X, Y), por lo que puede suprimirse. Sin embargo, al suprimir C también suprimimos sus asociaciones, con lo que perdemos la información sobre la cardinalidad de la asociación entre C y B. Para no perder esta información hay que dibujar la línea que representa a la asociación compuesta por A-C y C-B, es decir, la asociación implícita entre A y B pasando por C a través del atributo común X:



¿Cuál es la cardinalidad de esta asociación entre A y B? Viene definida por la tabla de cardinalidades de asociaciones compuestas que ya se ha mostrado anteriormente. Según esta tabla, el valor de a puede ser m ó 1, y el de b puede ser cualquiera (n, m, c, 1, 0). Si no hay condiciones adicionales que nos permitan restringir estos valores, con la sola información del diagrama no podemos hacerlo, por lo que tomaremos los más generales, es decir, a = m y b = n. Por tanto, el diagrama queda así:

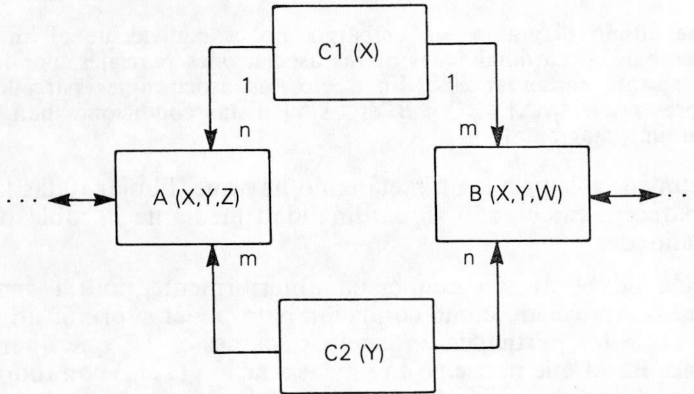


Suprimamos C:

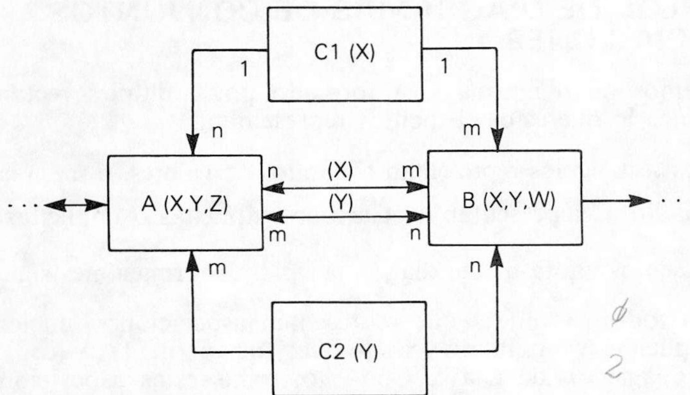


Ejemplo 3:

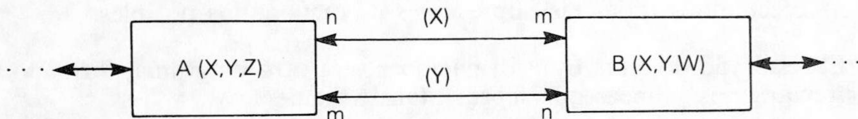
Supongamos que hay dos rectángulos como el C anterior, en vez de uno:



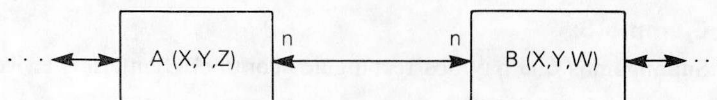
Representemos las asociaciones implícitas compuestas entre A y B según X (camino A-C1-B) y según Y (camino A-C2-B). Sus cardinalidades vienen dadas por la tabla de composición de cardinalidades. Tomando los valores más generales, el diagrama queda:



Como ya sabemos, C1 y C2 son redundantes, por lo que el diagrama se transforma en:



De acuerdo con lo explicado anteriormente sobre cardinalidad de asociaciones implícitas parciales, la cardinalidad de la asociación implícita total entre A y B puede ser n, c ó 0. Tomando los valores más generales, (n : n), queda el diagrama:



Este último diagrama, sin embargo, no es equivalente al anterior en el que figuraban las cardinalidades de las asociaciones parciales, por lo que puede ser interesante conservar éste. En efecto, las asociaciones parciales nos permiten expresar que: **A**[X]⊆**B**[X]; **B**[Y]⊆**A**[Y]. Estas condiciones han desaparecido en el último diagrama.

En resumen, al suprimir un rectángulo hay que dibujar todas las asociaciones que desaparecen, calculando su cardinalidad mediante la tabla de composición de cardinalidades.

La regla de Supresión enunciada anteriormente podría generalizarse. En efecto, hemos establecido como condición para poder suprimir un rectángulo que todos sus atributos participen en las asociaciones en las que interviene. Esto no es necesario. Basta que participe en una asociación (1 : m) con todos sus atributos para que ya se pueda suprimir, aunque participe en otras con solamente parte de sus atributos. Estas otras asociaciones habría que reflejarlas en el diagrama resultante, lo que puede ser complicado. Como además en la práctica suele bastar con la regla que hemos enunciado al principio, nos limitaremos a ella.

EJEMPLOS DE DIAGRAMAS DE CONJUNTOS Y ASOCIACIONES

Tomemos un diagrama C/A formado por múltiples rectángulos y líneas. Recordemos lo que estos elementos representan:

1) Los rectángulos representan conjuntos de valores, o sea, relaciones unarias.
2) Las líneas representan asociaciones explícitas o implícitas.

Apliquemos ahora a este diagrama las transformaciones siguientes:

1) En todas las líneas que representen asociaciones funcionales completas explícitas y no biunívocas (o sea, (n : 1), (m : 1) o (c : 1)) efectuamos la Propagación de Clave. Con ello, todas estas asociaciones pasan a ser implícitas.
2) En todas las líneas que representen asociaciones biunívocas (o sea, (1 : 1)) efectuamos la Agregación de los rectángulos.
3) Efectuamos todas las Supresiones de rectángulos posibles.

El resultado de estas transformaciones será otro diagrama, formado también por rectángulos y líneas que representan lo siguiente:

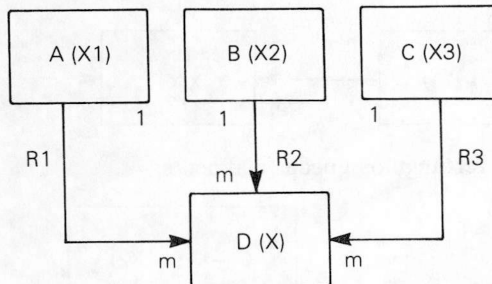
1) Los rectángulos representan ahora relaciones n-arias.
2) Las líneas representan asociaciones binarias, como antes, pero con la diferencia de que ahora no hay en el diagrama ninguna asociación

explícita que sea funcional completa. O sea, todas las asociaciones explícitas que quedan son (n : n), (n : m), (n : c), (m : m), (m : c) o (c : c).

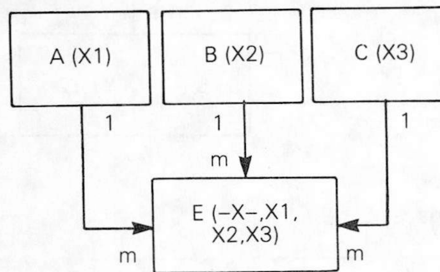
Veamos algunos ejemplos de transformación de diagramas.

Ejemplo 1:

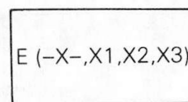
Sea el diagrama C/A:



Propagando las claves a través de R1, R2 y R3, queda:

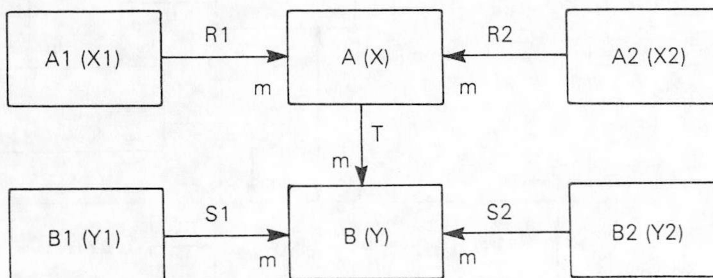


Suprimiendo rectángulos queda:

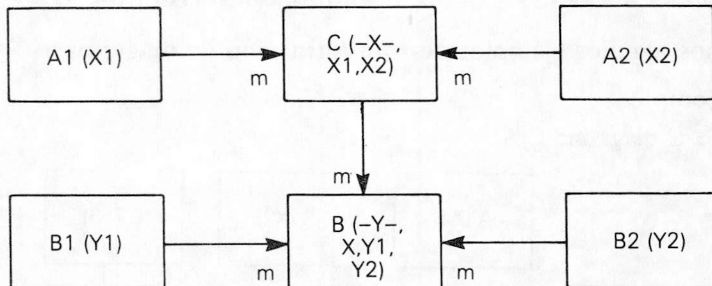


Ejemplo 2:

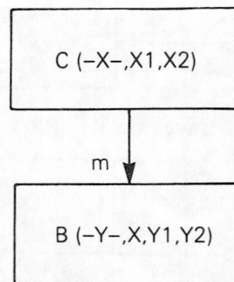
Sea el diagrama C/A:



Propagando las claves queda:

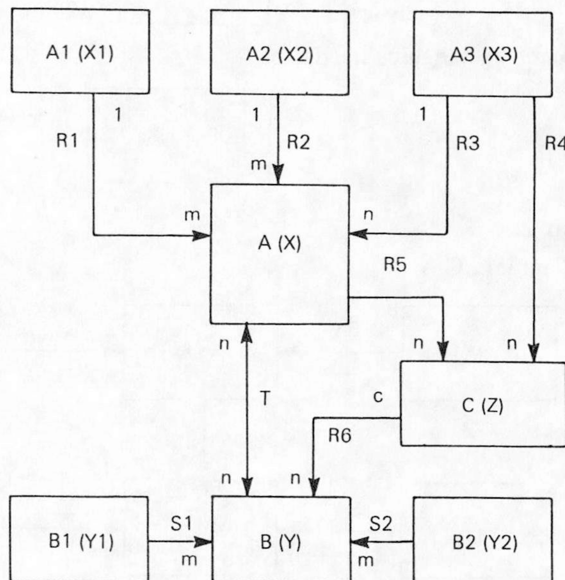


Suprimiendo rectángulos queda finalmente:

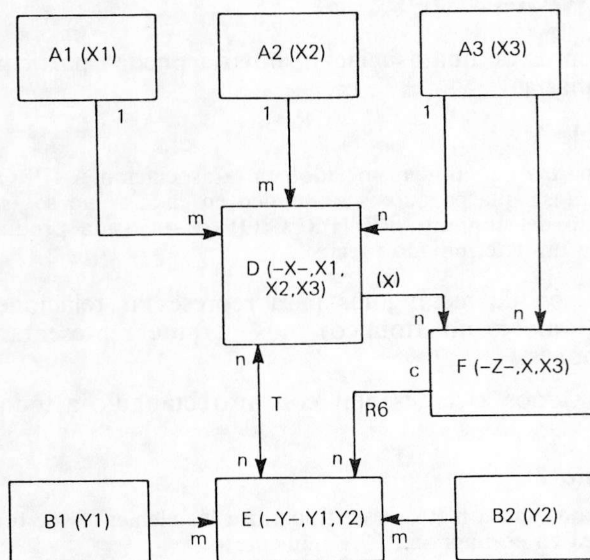


Ejemplo 3:

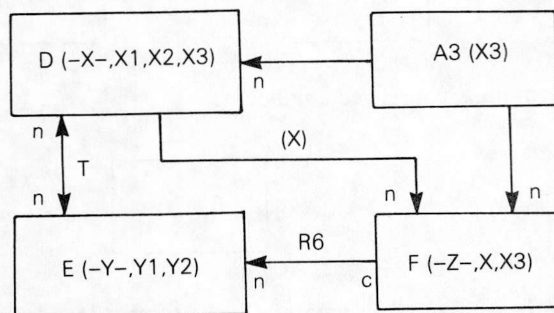
Sea el diagrama C/A:



Propagando las claves queda:



Suprimiendo rectángulos queda finalmente:



Este diagrama, como vemos, tiene 4 rectángulos (D, E, F y A3), dos asociaciones explícitas, una, T, que es (n : n), y otra, R6, que es (n : c), y tres asociaciones implícitas, de las que dos son totales y la otra parcial.

El esquema de diseño será:

Relaciones: A3(X3), D(–X–, X1, X2, X3), E(–Y–, Y1, Y2), F(–Z–, X, X3), T(X, Y), R6(–Y–, Z).

Condiciones: **D**[X3] ⊆ A3(X3); **F**[X3] ⊆ A3(X3); **F**[X] ⊆ **D**[X].

(Obsérvese que todo valor de X en F, además de existir en D, tiene que estar en D no repetido, pero esto ya se cumple por ser X clave en D, por lo que sería redundante incluir esta condición.)

5

REPRESENTACION DE ASOCIACIONES ENTRE ASOCIACIONES

Los elementos de una asociación binaria pueden participar a su vez en otra asociación binaria.

Ejemplo 1:

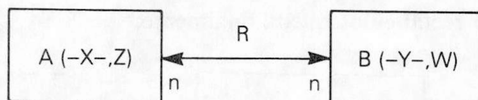
Cada pareja de valores asociados en la asociación A (PRODUCTOS, ZONAS), que expresa qué productos se venden en cada zona, se asocia a su vez con un elemento del conjunto PR (PRECIO) (o sea, cada producto en cada zona, se vende a un determinado precio).

La utilización de rectángulos para representar relaciones n-arias, es decir, conjuntos de valores no atómicos, nos permite representar estas asociaciones entre asociaciones.

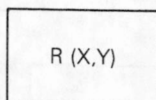
Para ello vamos a representar con un rectángulo a toda asociación binaria explícita.

Ejemplo 2:

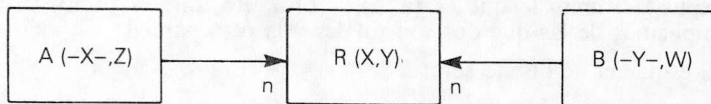
Sea una asociación R de este tipo entre los elementos de otras dos, A y B, como se indica en el diagrama C/A siguiente:



Si representamos con el rectángulo:



al conjunto de parejas de valores (X, Y) (o sea, las claves de A y B), asociados en R, el diagrama anterior se transforma en:

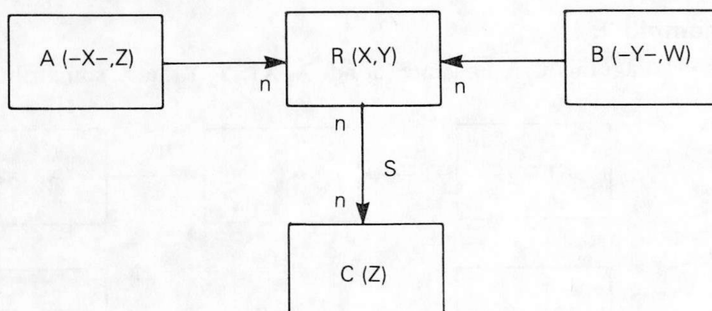


Vemos que la asociación explícita R, de cardinalidad (n : n), se ha transformado así en un rectángulo y en dos asociaciones implícitas funcionales completas (o sea, de tipo (1 : n)).

Este último diagrama nos permite ya representar otras asociaciones en las que participen las tuplas de R.

Ejemplo 3:

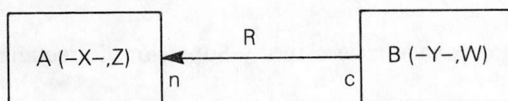
Sí hay una asociación explícita entre los elementos de R y las de otro rectángulo C(Z), tendríamos el diagrama siguiente:



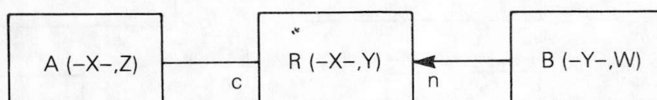
Obsérvese que si alguna de las dos asociaciones implícitas, de A con R o de B con R, es de cardinalidad (1 : c), el atributo común debe ser clave en R.

Ejemplo 4:

Así, si R es (n : c):



resultará que X debe ser clave de R (por la Regla de Definición de Claves):



DIAGRAMAS RE/R

Supongamos que en un diagrama C/A transformamos en asociaciones binarias funcionales completas todas las asociaciones explícitas que no lo sean, mediante el procedimiento descrito de representarlas con un rectángulo. En este diagrama todas las asociaciones serán implícitas. A un diagrama con esta característica lo llamaremos «relacional».

Si le aplicamos las Reglas de Definición y Propagación de Claves, y Agregación y Supresión de Rectángulos, tantas veces como sea posible, obtendremos finalmente un diagrama al que llamaremos diagrama RE/R (diagrama Relacional de Entidades y otras Relaciones).

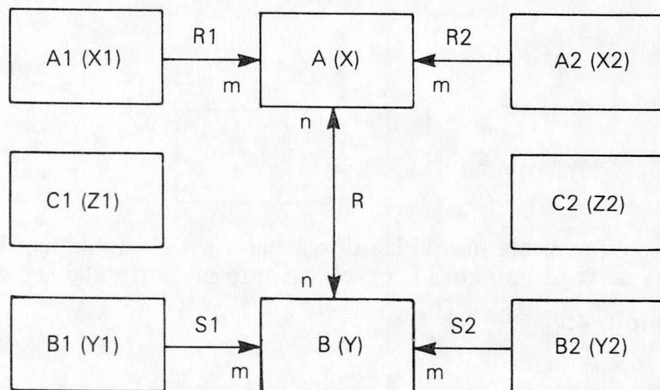
A partir de un diagrama RE/R el esquema relacional de diseño se obtiene directamente, pues cada rectángulo da lugar a una relación del esquema, con sus

claves. Además, las asociaciones implícitas nos darán otras condiciones de integridad entre las relaciones (integridad de referencia, por ejemplo).

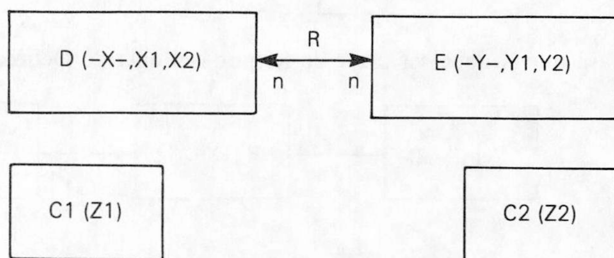
Ejercitemos estas ideas con algunos ejemplos.

Ejemplo 1:

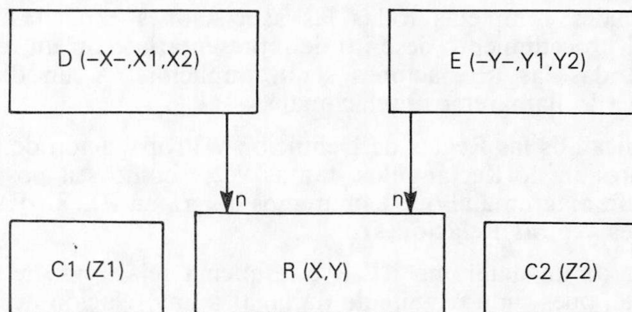
Sea el diagrama C/A siguiente (donde X, X1, Y, Y1, etc., son atributos simples):



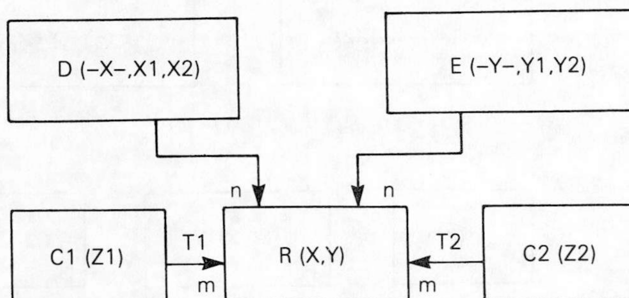
Mediante Propagación de claves y Supresión de rectángulos se transforma en:



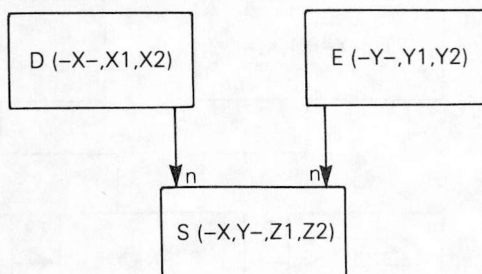
Transformemos todas las asociaciones explícitas en implícitas:



Supongamos que entre los elementos de R y los de C1 y C2 hay dos asociaciones, T1 y T2. Representémoslas en el diagrama:



Propagando claves y suprimiendo rectángulos llegamos finalmente al diagrama RE/R:



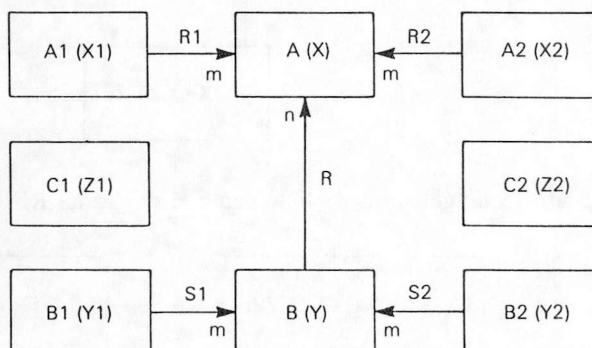
El correspondiente esquema relacional de diseño será:

Relaciones: D(–X–, X1, X2), E(–Y–, Y1, Y2), S(–X, Y–, Z1, Z2).

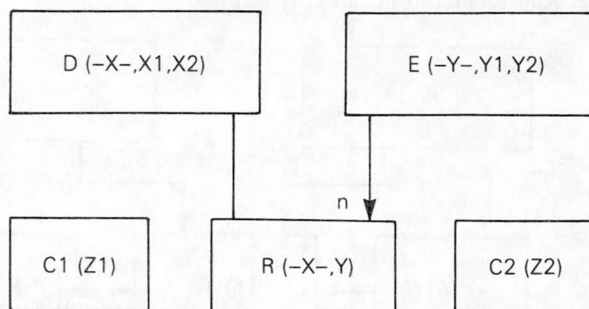
Otras condiciones: **S**[X]⊆**D**[X]; **S**[Y]⊆**E**[Y].

Ejemplo 2:

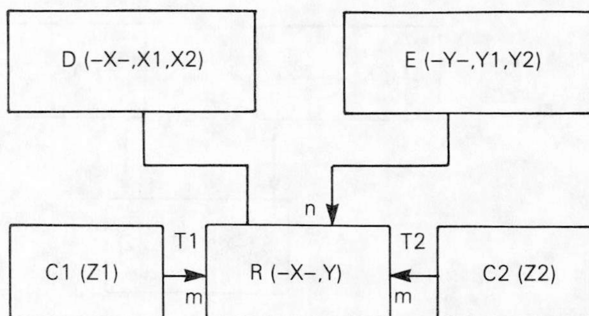
Supongamos ahora el mismo ejemplo anterior, pero con R de cardinalidad (n : 1). Diagrama C/A de partida:



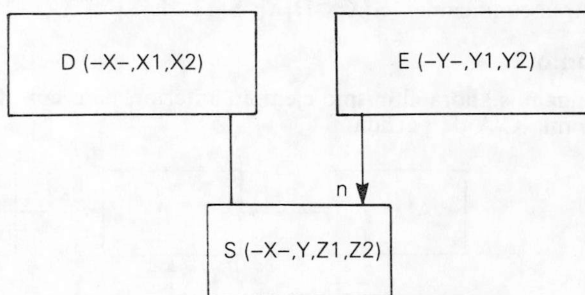
Transformemos todas las asociaciones explícitas en implícitas:



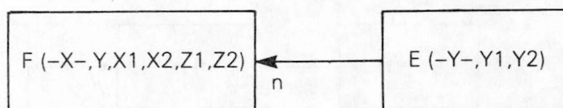
Si entre los elementos de R y los de C1 y C2 hay, como antes, asociaciones (m : 1):



Propagando claves y suprimiendo rectángulos:



Agregando rectángulos queda finalmente el diagrama RE/R:



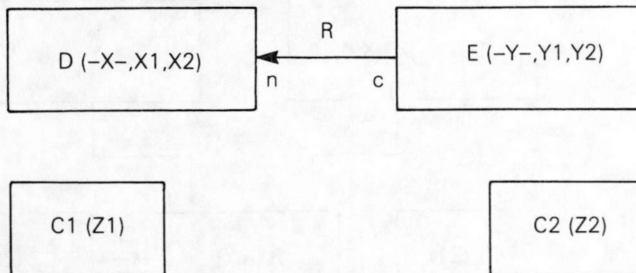
El correspondiente esquema de diseño será:

F(–X–, Y, X1, X2, Z1, Z2); E(–Y–, Y1, Y2); F[Y]⊆E[Y].

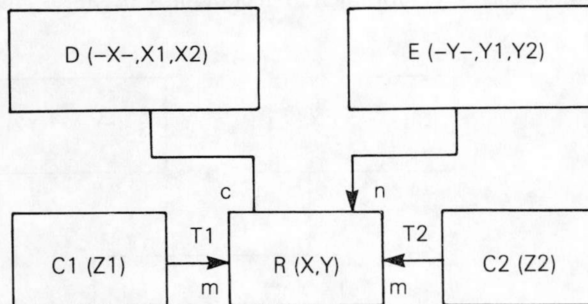
Ejemplo 3:

Supongamos otra vez el mismo ejemplo anterior, pero con R de cardinalidad (n : c).

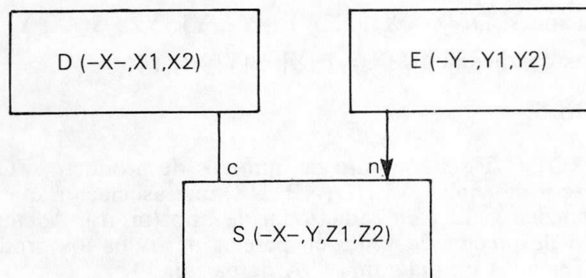
El diagrama C/A de partida se transforma en:



Lo completamos:



Por Definición de Claves, y propagando claves y suprimiendo rectángulos llegamos finalmente al diagrama RE/R:



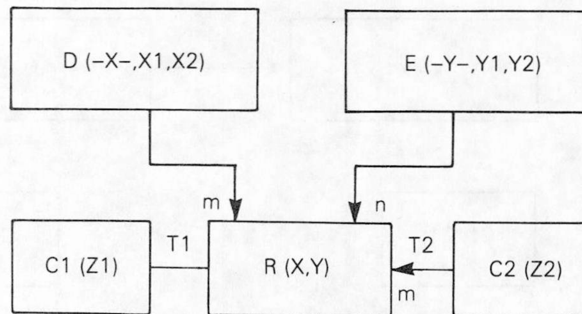
El correspondiente diagrama relacional de diseño será:

Relaciones: D(–X–, X1, X2); E(–Y–, Y1, Y2); S(–X–, Y, Z1, Z2).

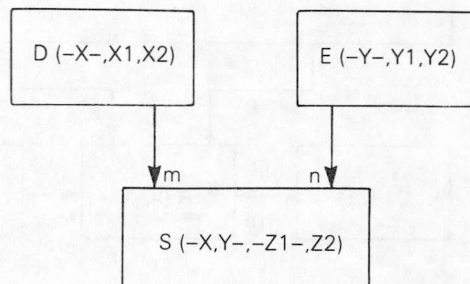
Otras condiciones: S[X]⊆**D**[X]; **S**[Y]⊆E[Y].

Ejemplo 4:

Tomemos de nuevo el mismo ejemplo, pero ahora con R de cardinalidad (n : m), T1 de cardinalidad (1 : 1) y T2 (m : 1):



Propagando claves y suprimiendo rectángulos llegamos finalmente al diagrama RE/R:



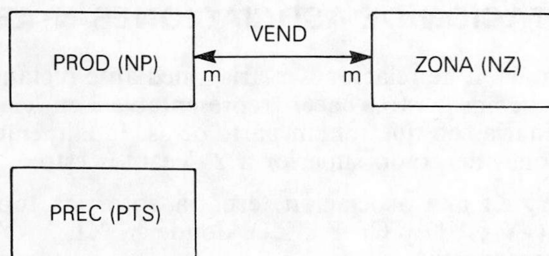
El correspondiente diagrama relacional de diseño será:

Relaciones: D(–X–, X1, X2); E(–Y–, Y1, Y2); S(–X, Y–, –Z–, Z1, Z2).

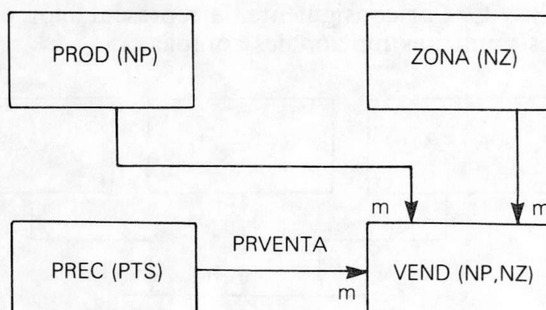
Otras condiciones: *S*[X] = **D**[X]; **S**[Y]⊆**E**[Y].

Ejemplo 5:

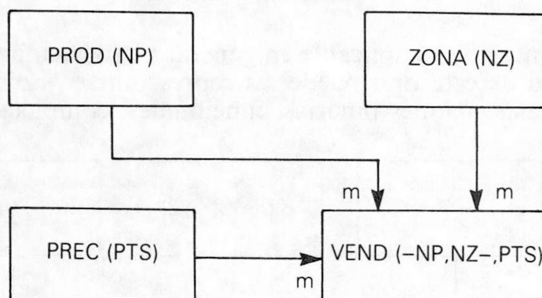
Sean PROD(NP) el conjunto de números de producto, ZONA(NZ) el conjunto de números de zona y VEND(NP, NZ) una asociación que significa qué productos se pueden vender en cada zona, de tipo (m : m). Además, PREC(PTS) es el conjunto de precios de venta en pesetas de todos los productos en las diversas zonas. Tenemos un diagrama C/A de partida:



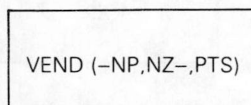
Asociemos los precios con los productos y zonas:



Propagando claves obtenemos el diagrama RE/R:



Suprimiendo rectángulos queda finalmente el diagrama RE/R:



El esquema de diseño es la relación:

VEND(–NP, NZ–, PTS)

sin otras condiciones adicionales.

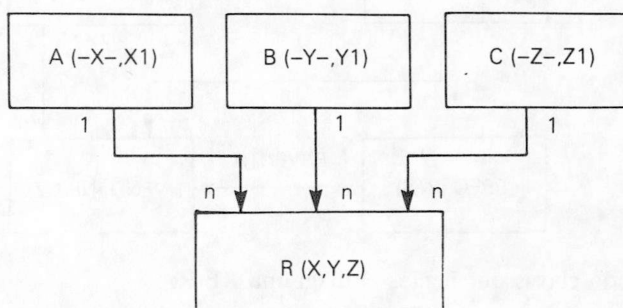
5

REPRESENTACION DE ASOCIACIONES N-ARIAS

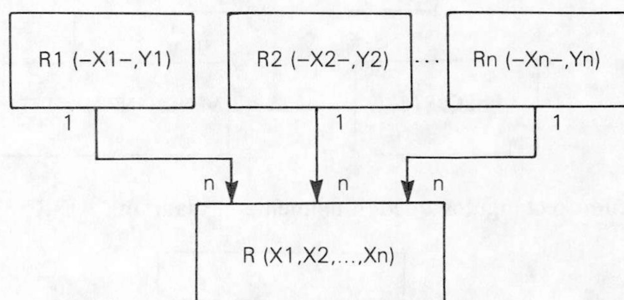
La representación de relaciones n-arias mediante rectángulos nos ha permitido, como ya hemos visto, hacer representables en el diagrama RE/R las asociaciones binarias en que toman parte otras. Igualmente nos permiten representar asociaciones de grado superior a 2. Veamos esto.

Sea R(X, Y, Z) una asociación ternaria entre las tuplas de los conjuntos A(–X–, X1), B(–Y–, Y1) y C(–Z–, Z1), donde X, X1, Y, Y1, etc., son atributos posiblemente compuestos.

Representemos con un rectángulo el conjunto de tuplas de la relación R. Evidentemente, todos los valores de X, Y y Z existentes en R, deben existir también en A, B, y C. Por consiguiente, la representación en el diagrama tendrá tres asociaciones implícitas funcionales completas:



Esta representación es aplicable en general a asociaciones de cualquier grado. Toda asociación de este tipo puede así representarse gráficamente mediante un rectángulo y n asociaciones binarias, funcionales, completas e implícitas:

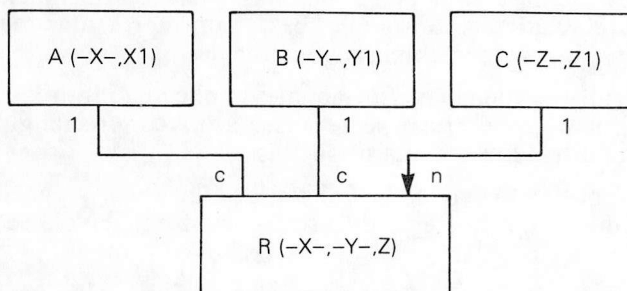


La cardinalidad en los extremos de las líneas que acaban en R, podrá ser, en general, una combinación cualquiera de n, m, c y 1, por ejemplo: (n : n : ... : n), (1 : m : n : ... : n), (m : c : 1 : ... : n), etc.

Naturalmente, siguen siendo aplicables las reglas de transformación que ya hemos usado repetidas veces.

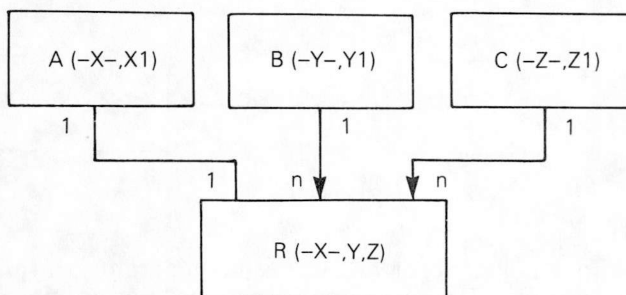
Ejemplo 1:

Si tenemos un diagrama con cardinalidades (c : c : n), como el siguiente, se deduce que X e Y deben ser claves en R:

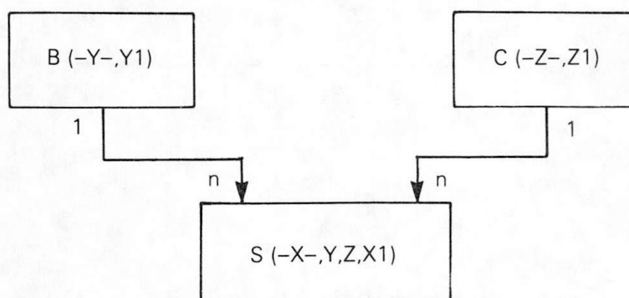


Ejemplo 2:

Si la cardinalidad fuera (1 : n : n) podríamos agregar rectángulos. Sea:



se transforma en:



(con S = A * R).

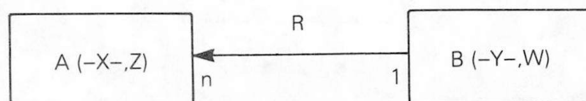
5

REDUNDANCIA DE LA REGLA DE PROPAGACION DE CLAVES

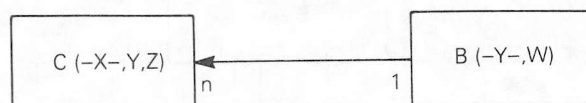
La Regla de Propagación de Claves no es estrictamente necesaria, pues puede substituirse por una combinación de las reglas de Definición de Claves y Agregación de Rectángulos. Para ello, basta con representar mediante rectángulos también las asociaciones binarias funcionales.

Por consiguiente, todas las asociaciones explícitas, tanto las binarias, funcionales o no, como las n-arias, se representarán con rectángulos, ligados por asociaciones binarias funcionales implícitas.

Veamos cómo las Reglas de Definición de Claves y Agregación de Rectángulos hacen redundante a la Regla de Propagación de Claves. Sea el diagrama:

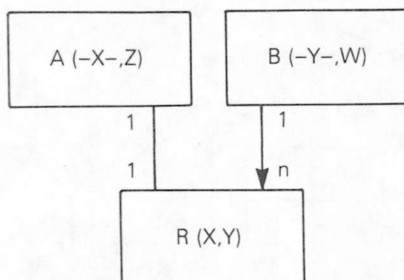


Aplicando la Regla de Propagación de Claves tendremos que es equivalente a:

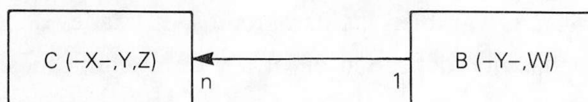


(donde C = A * R).

Veamos ahora cómo llegar a este mismo diagrama sin aplicar la Regla de Propagación de Claves. Para ello transformamos la asociación explícita R en dos implícitas, representándola con un rectángulo:



En este dibujo, aplicando la Regla de Definición de Claves, se deduce que X es clave en R(X, Y), o sea: R(–X–, Y). Además, agregando los rectángulos A y R queda:



con C(X, Y, Z) = A(X, Y) * R(X, Y). Vemos que hemos llegado al mismo diagrama que anteriormente.

DEFINICION DE ENTIDADES

Sea un diagrama RE/R, obtenido aplicando las reglas de transformación tantas veces como sea posible. Estará formado por los siguientes elementos:

1) Rectángulos que representan relaciones n-arias (n = 1, 2, 3, ...).
2) Líneas entre ellos que representan asociaciones implícitas.

Diremos que cada rectángulo representa un Tipo de Entidad o simplemente Entidad (con mayúscula). Las tuplas de la relación representada por un rectángulo representan ocurrencias de ese Tipo de Entidad o simplemente entidades (con minúscula).

Diremos que una Entidad B depende de otra A si toda ocurrencia de B exige la existencia previa de una ocurrencia en A. Es decir, si entre ellas existe una asociación cuya cardinalidad no incluye el valor 0 en A. O también, si la cardinalidad de la asociación entre A y B es (a : b), donde a es 1 ó m y b puede ser cualquier valor (n, m, c, 1).

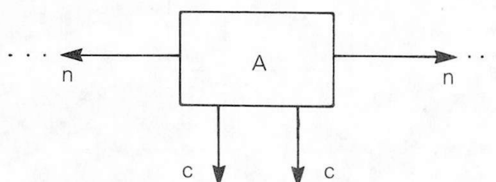
Cabe distinguir tres clases de Tipos de Entidad:

1) *Entidades fuertes*. También las llamaremos Entidades independientes o simplemente Entidades.

   Una Entidad es independiente cuando no depende de ninguna otra. Es decir, cuando todas las asociaciones entre ella y otras tienen en estas últimas una cardinalidad distinta de 1 ó m.

Ejemplo 1:

La Entidad representada por el rectángulo A del diagrama siguiente, puesto que todas sus asociaciones terminan en una cardinalidad que incluye el 0:

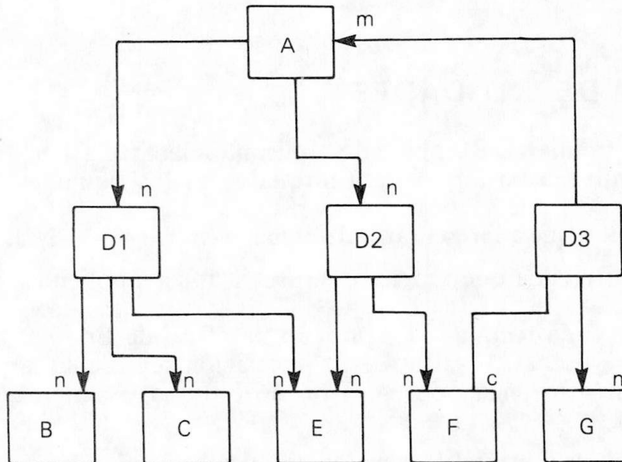


2) *Entidades débiles.* También las llamaremos Entidades dependientes.

Son aquellas que dependen de otra y de ninguna más.

Ejemplo 2:

Son débiles las entidades A, D1, D2, D3, B, C y G del diagrama:

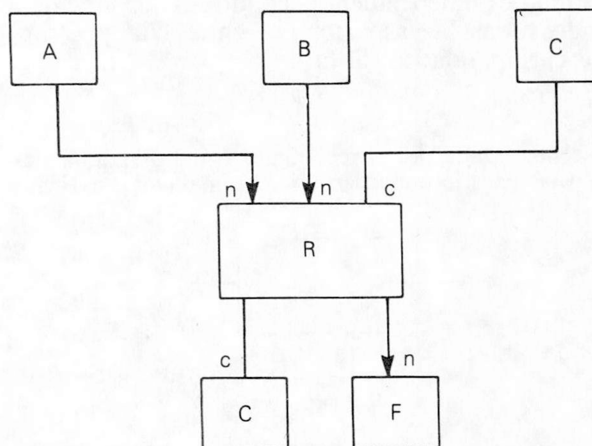


Dos entidades pueden ser interdependientes si una depende de la otra, y viceversa. Por ejemplo, las Entidades A y D3 del diagrama anterior.

3) *Entidades asociativas.* Son las que dependen de otras dos o más.

Ejemplo 3:

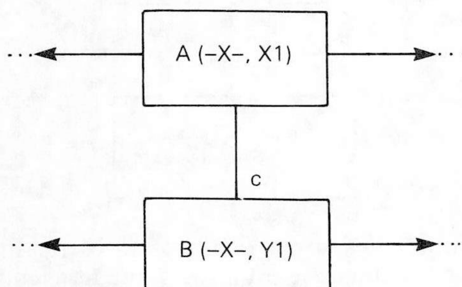
La Entidad R siguiente es asociativa:



Entre las Entidades que dependen de otras hay un caso que merece ser destacado y que permite definir a una de ellas como subconjunto o subtipo de la otra.

Sea una Entidad A con una clave X, y otra B asociada con A a través del atributo X con una cardinalidad (1 : c). Diremos en este caso que B es una Entidad subordinada a A, o que B es un subtipo de A.

El correspondiente diagrama sería:



Ejemplo 4:

Sean las Entidades y relaciones siguientes:

1) Empleados: EMP(–#E–, NE)
   #E: Identificador de empleado
   NE: Nombre de empleado

2) Vendedores: VEN(–#E–, CO)
   CO: Comisión

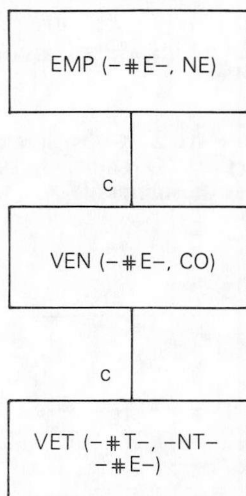
3) Vendedores con territorio:
   VET(–#T–, –NT–, –#E–)
   #T: Identificador de territorio
   NT: Nombre de territorio

Supongamos que se cumplen las condiciones siguientes:

- Todo vendedor es un empleado. Es decir, la Entidad VEN es un subtipo o clase de la Entidad EMP.
- Todos los territorios tienen un vendedor asignado y sólo uno. Hay vendedores que no tienen territorio porque actúan a través de concesionarios.

Estas condiciones nos dan el diagrama:

5



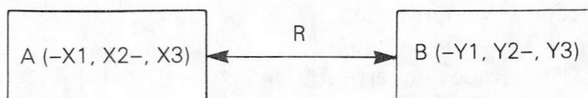
Todas estas definiciones sobre distintos tipos de Entidades suelen tener un reflejo en aspectos semánticos de éstas. Aunque no siempre tenga que ser así, puesto que las definiciones dadas se basan en un aspecto puramente formal, son sin embargo éstas una buena ayuda para perfilar y precisar el significado intuitivo de las Entidades.

ASOCIACIONES EXPLICITAS PARCIALES

Diremos que una asociación explícita es parcial cuando no intervienen en ella todos los atributos de las claves de los rectángulos asociados.

Cuando se dibuja un diagrama de diseño hay que tener cuidado de no incluir asociaciones explícitas parciales.

Recordemos el significado de una asociación explícita. Sea el diagrama:



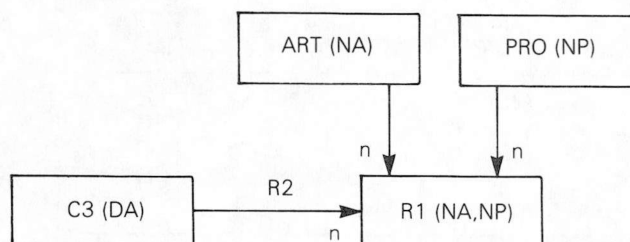
Este diagrama significa que entre las tuplas de A y B existe una asociación explícita R. Puesto que las tuplas están identificadas por las claves, esto es equivalente a decir que R es una asociación entre las claves de A y B. Esto quedaría desvirtuado si R sólo existiera entre parte de los atributos de las claves, es decir, si R fuera una asociación parcial. Si éste fuera el caso, el diagrama anterior sería incorrecto. Habría que dibujar R entre otros rectángulos, distintos de A y B, que tuvieran como claves a los atributos de A y B implicados en R. Si no se hace así aparecerán dependencias transitivas en el diseño final, por lo que éste no estará normalizado.

Ejemplo:

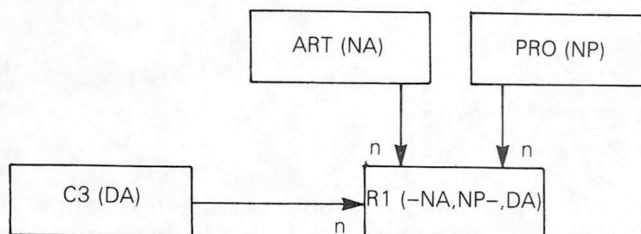
Tomemos los siguientes conjuntos y asociaciones:

PRO(NP): Números identificadores de Proveedores.
ART(NA): Números identificadores de Artículos.
C3(DA): Conjunto de descripciones identificadoras de Artículos.
R1: Pedidos (Asociación que expresa «Artículos pedidos a Proveedores»).

Supongamos que en cada pedido, además del identificador del Artículo y del Proveedor, figura también la descripción del Artículo. La asociación entre ésta y el pedido la llamaremos R2. Podemos representar esto en el diagrama:



En este diagrama la asociación explícita R2 es parcial, pues la descriición de un artículo, DA, va asociada realmente a su identificador, NA, y no a la clave completa del pedido, (NA, NP). Este diagrama es incorrecto y producirá un diseño no normalizado. En efecto, aplicándole las reglas de transformación obtenemos:

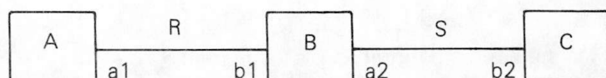


En el esquema de diseño figurará por tanto la relación: R1 (–NA, NP–, DA). En esta relación se verifica las dependencias NA→DA, y DA→NA, cuyos antecedentes no son claves, y por tanto no está normalizada.

En conclusión, al dibujar el diagrama hay que estar atento a las asociaciones explícitas entre rectángulos con claves compuestas para asegurarse de que no son parciales.

PRODUCTO DE ASOCIACIONES

Sea una asociación entre dos rectángulos A y B y otra entre B y C, explícitas, y de cardinalidades cualesquiera:



Definamos ahora una asociación ternaria, T, entre A, B y C, asociando a cada valor de A los de B que le corresponden en R, y a éstos los de C que le corresponden en S.

Ejemplo 1:

Supongamos los valores siguientes:

Valores de A	*R (A, B)*	*Valores de B*	*S (B, C)*	*Valores de C*
1	(2, a)	a	(b, A)	A
2	(2, b)	b	(b, C)	B
3	(4, b)	c	(d, C)	C
4	(4, c)	d		D
5				

La Asociación T sería:

T	A	B	C
	2	b	A
	2	b	C
	4	b	A
	4	b	C

Evidentemente, $T = R * S$.

Llamaremos producto de dos asociaciones, R y S, con un rectángulo común, B, como en el diagrama anterior, a otra asociación, P, entre los elementos de A y C, que asocia a cada elemento de A los de C que, según S, están asociados a los de B que están a su vez asociados, según R, al elemento considerado de A. Es decir, asociamos a cada elemento de A los de C que se obtienen recorriendo el camino A-R-B-S-C del diagrama. O lo que es igual: $P = T[A, C]$.

Ejemplo 2:

Para los valores del ejemplo anterior tenemos:

P	A	C
	2	A
	2	C
	4	A
	4	C

La cardinalidad de P está condicionada por la de los factores, R y S. Así en el extremo de C la cardinalidad de P puede tomar los valores indicados en la tabla siguiente:

Cardinalidad del Producto

b2: / b1:	n	m	c	l
n	n, c, 0	n, c	n, c, 0	n, c
m	n, m, c, 1, 0	m, 1	n, m, c, 1, 0	m, 1
c	n, c, 0	n, c	c, 0	c
1	n, m, c, 1, 0	m, 1	c, 1, 0	1

Ejemplo 3:

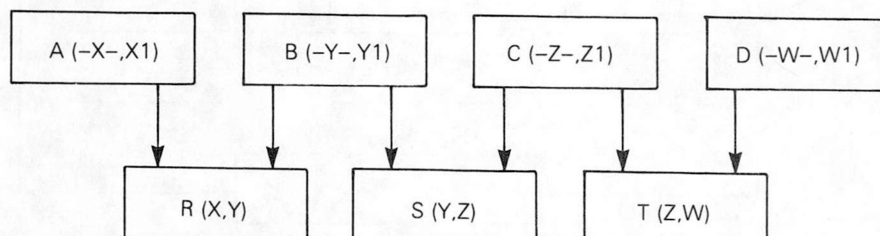
Así, si R es (1 : m) y S es (c : n), la cardinalidad del producto de ambas vendrá dada por c · 1 = c y m · n = (n, m, c, 1 ó 0). Tomando los valores más generales, la cardinalidad sería (c : n).

Como la asociación P es obtenible a partir de R y S, es redundante con ellas, por lo que no deberá figurar en el esquema de diseño.

La operación de multiplicar asociaciones puede definirse con más de dos factores si recorremos un camino del diagrama RE/R de más de dos tramos.

Ejemplo 4:

Sean las asociaciones implícitas siguientes:



Si recorremos el camino A-R-B-S-C-T-D, asociamos a cada elemento de A todos los de B que le correspondan en R, a éstos todos los C que le correspondan en S, etc. Obtendremos una relación de asociación, Q:

$$Q(X, Y, Z, W) = \mathbf{A}[X] * R(X, Y) * \mathbf{B}[Y] * S(Y, Z) * \mathbf{C}[Z] * T(Z, W) * \mathbf{D}[W]$$

Como $(\mathbf{A}[X] * R(X, Y) = R(X, Y))$ y $(\mathbf{B}[Y] * S(Y, Z) = S(Y, Z))$, etc. queda:

$$Q(X, Y, Z, W) = R(X, Y) * S(Y, Z) * T(Z, W)$$

Esta relación, o cualquiera de sus proyecciones, es redundante.

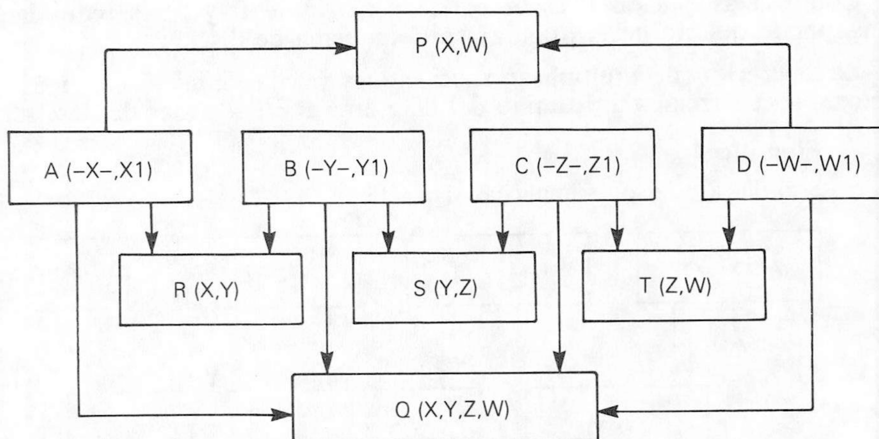
Siempre que en un diagrama hay asociaciones redundantes, su dibujo presenta bucles (prescindiendo de las flechas, pues éstas no indican sentido, sino sólo cardinalidad). El recíproco no es cierto.

Puede haber bucles sin que haya redundancia de asociaciones. Esto depende del significado de éstas. Si el bucle se debe a que hay alguna asociación redundante, hay que romper el bucle, si es posible, eliminándola. A veces, dentro de un bucle redundante puede eliminarse una u otra asociación, lo que obliga a elegir una de ellas. Para ello puede ser conveniente dibujar varios diagramas, con las diversas alternativas, y elegir aquel que produzca el mejor diseño final. Esto será más fácil de realizar si el proceso de diseño se apoya en alguna herramienta mecanizada. En este tipo de decisiones, la experiencia y buen juicio del analista serán el mejor consejero.

Veamos algunos ejemplos.

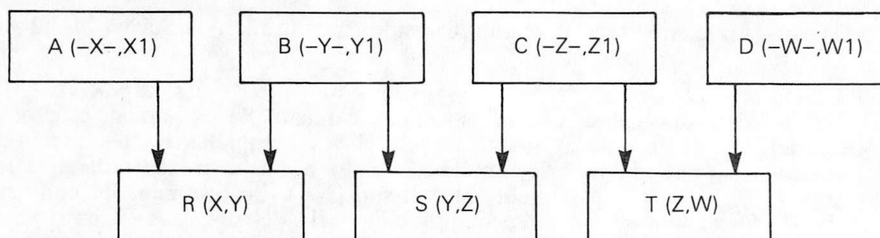
Ejemplo 5

Sea el diagrama RE/R siguiente:



Vemos que hay varios bucles. Por ejemplo, (A-R-B-S-C-T-D-P-A), (A-R-B-S-C-T-D-Q-A), (B-S-C-Q-B), etc. Supongamos que $Q = R(X, Y) * S(Y, Z) * T(Z, W)$. Por consiguiente, al ir de A a D por el camino (A-R-B-S-C-T-D) se obtiene el mismo resultado que por el camino (A-Q-D). Supongamos que ocurre lo mismo yendo por el camino (A-P-D). Es decir, que se obtienen los mismos valores de W asociados a un valor cualquiera de X yendo de A a D por cualquiera de estos caminos. En resumen, los bucles son debidos a que las asociaciones Q y P son redundantes con R, S y T.

Eliminándolas queda el siguiente diagrama:



Ejemplo 6:

Sean los siguientes conjuntos:

PRO(NP): Números identificadores de Proveedores.
ART(NA): Números identificadores de Artículos.
CLI(NC): Números identificadores de Clientes.
C1(DA): Descripciones de Artículos.

Y sean las siguientes asociaciones entre ellos:

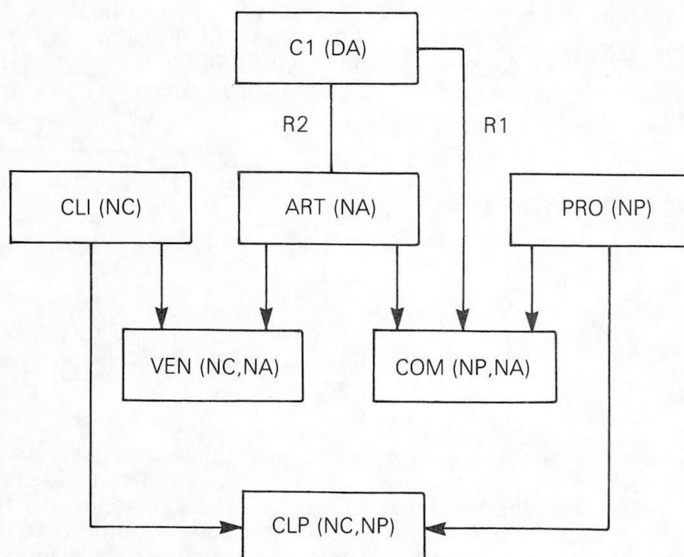
COM = Pedidos de Compras (conjunto de Artículos que hemos pedido a nuestros Proveedores).

VEN: Pedidos de Ventas (conjunto de Artículos que nos han pedido nuestros Clientes).

CLP: Conjunto de Clientes que nos han comprado Artículos suministrados por un determinado Proveedor.

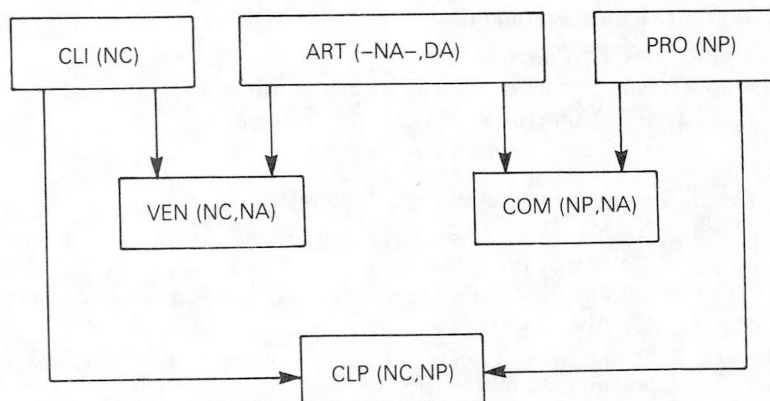
R1: Descripciones de los Artículos en los Pedidos de Compras.

Supongamos que el diagrama de partida es:

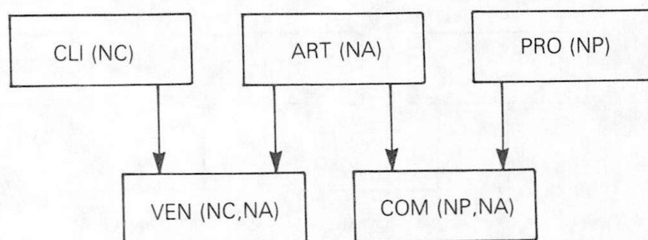


Hay varios bucles en este diagrama. Consideremos primero el bucle (C1-R2-ART-COM-R1-C1).

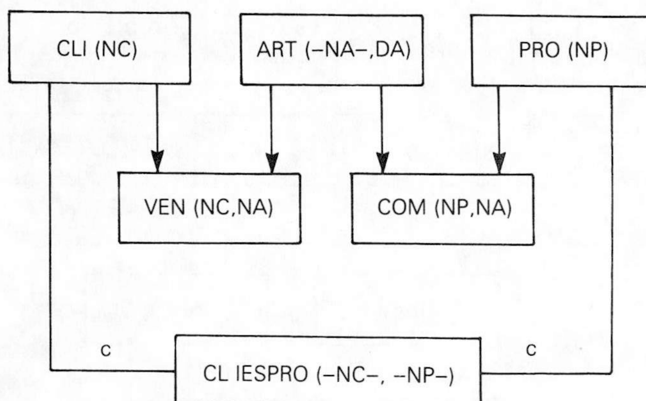
R1 (NP, NA, DA) y R2 (–NA–, –DA–) son redundantes, pues R2 = **R1**[NA, DA]. La causa de este bucle es que la asociación explícita R1 es parcial, es decir, no debieran intervenir en ella realmente las claves completas de los rectángulos participantes, pues la descripción del Artículo no debe ir asociada al Pedido (clave: (NP, NA)), sino solamente al Artículo (NA). En resumen, R1 contiene la misma información que R2 y podemos eliminarla del diagrama. Como además R2 es de cardinalidad (1 : 1) se pueden agregar los rectángulos C1 y ART. Nos queda:



Evidentemente, los clientes que compran los artículos suministrados por un proveedor determinado se obtienen entrando en el diagrama por proveedor y recorriendo el camino PRO-COM-ART-VEN-CLI, o también, entrando por proveedor y recorriendo el camino PRO-CLP-CLI. Es decir, CLP es redundante con VEN y COM. O dicho de otro modo: CLP(NC, NP) = **P**(NC, NP) (VEN * COM). Eliminando CLP, el diagrama sin redundancias será:



Supongamos ahora que es posible que algunas empresas que son Proveedores nuestros también sean Clientes. Esto lo reflejaremos en el diagrama mediante una asociación entre los conjuntos CLI y PRO. La llamaremos CLIESPRO («Cliente es Proveedor») y nos asociará el identificador de un Proveedor con el de un Cliente cuando correspondan a una misma empresa. El diagrama quedará:



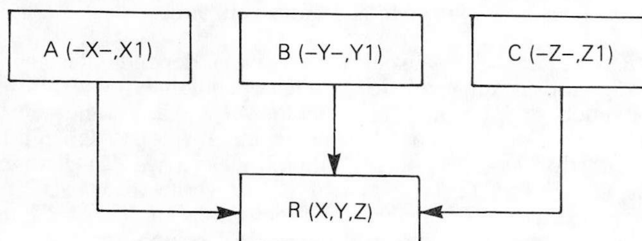
En este diagrama hay bucles, pero no redundancias. Es interesante observar que puede aparecer aquí una condición de integridad que no es expresable en el diagrama, ni tampoco como una dependencia de las estudiadas, que es la siguiente: probablemente, una empresa no nos comprará Artículos que ella misma nos suministre. Es decir, que (VEN * COM * CLIESPRO) es una relación vacía. Es posible, sin embargo, que en alguna ocasión se viole esta condición. Por ejemplo, si un Proveedor se quedare sin existencias de un Artículo que nos vendió previamente y los necesitare urgentemente, podría decidir recomprárnoslos. Esta consideración puede hacer aconsejable no incluir esta condición en el esquema final de diseño.

En conclusión, si al dibujar el diagrama aparecen bucles, hay que analizarlos para determinar si se deben a la presencia de asociaciones redundantes o no, y en caso afirmativo suprimirlas si es posible. Si no se suprimen, el diseño final que se obtenga estará probablemente no normalizado (posiblemente contendrá dependencias transitivas).

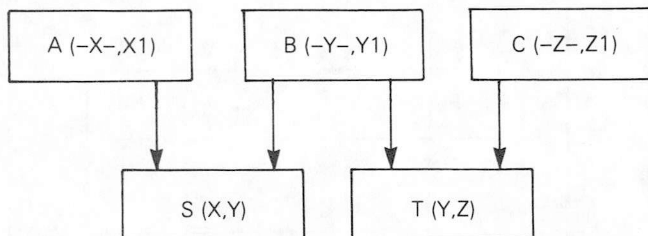
DEPENDENCIAS PLURALES

Puede ocurrir también que una asociacón ternaria (o de grado superior) se pueda descomponer en otras.

Consideremos el diagrama:



Sean S=**R**[X, Y] y T=**R**[Y, Z]. Si R es descomponible en S y T, será R=S * T. Entonces, el diagrama anterior sería equivalente a:



Ejemplo 1:

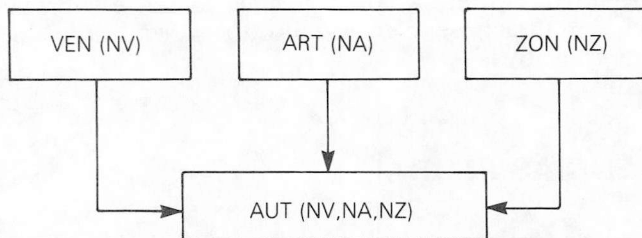
Supongamos los conjuntos de valores siguientes:

VEN(NV): Números identificadores de Vendedores.

ART(NA): Números identificadores de Artículos.

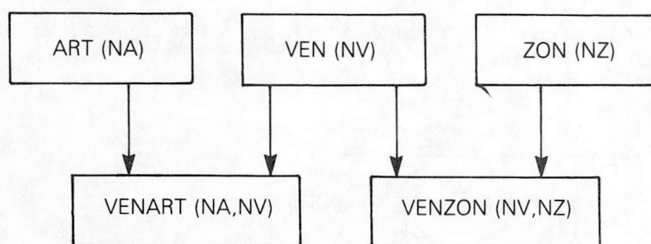
ZON(NZ): Números identificadores de las Zonas geográficas en que se divide el país a efectos de gestión del mercado.

Sea una asociación ternaria, AUT(NV, NA, NZ), que indica qué vendedores están autorizados a vender determinados artículos en las distintas zonas. El diagrama RE/R será:



La relación VENART=**AUT**[NV, NA] nos dice cuáles son los artículos que cada vendedor está autorizado a vender. La relación VENZON=**AUT**[NV, NZ] nos dice cuáles son las zonas asignadas a cada vendedor.

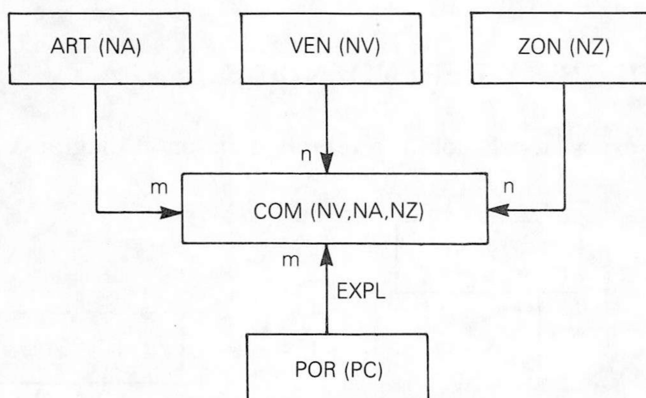
Supongamos que estas dos asociaciones son independientes, es decir, que los artículos asignados a un vendedor son independientes de sus zonas. Así, si por ejemplo el vendedor V1 vende los artículos A1 y A2, y vende en las zonas Z1 y Z2, todas las combinaciones posibles de artículos y zonas deben estar autorizadas para este vendedor: (A1, Z1), (A1, Z2), (A2, Z1) y (A2, Z2). Esto equivale a decir que en la relación AUT(NV, NA, NZ) se verifican las DPs: (NV→»NZ) y (NV→»NA). Por tanto, AUT será descomponible en VENART y VENZON, y será semánticamente más correcto dibujar el diagrama así:



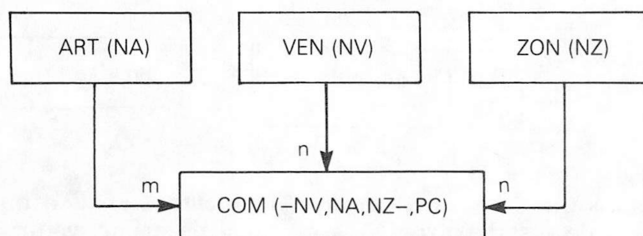
Si una asociación es de grado superior a 3, pueden presentarse DPs embebidas.

Ejemplo 2:

Supongamos que en el ejemplo anterior cada vendedor cobra un porcentaje de comisión que depende de cada artículo y zona. El correspondiente diagrama RE/R será:



Propagando la clave queda:



Como en el caso anterior, si suponemos que los artículos asignados a un vendedor son independientes de las zonas en que éste trabaja, la asociación ternaria entre (NV, NA, NZ), se compone de dos binarias independientes. Así, si el vendedor V1 puede vender los artículos A1 y A2, y en las zonas Z1 y Z2, entonces este

vendedor podrá recibir comisiones por todas las combinaciones de los artículos y zonas en que trabaja, es decir:

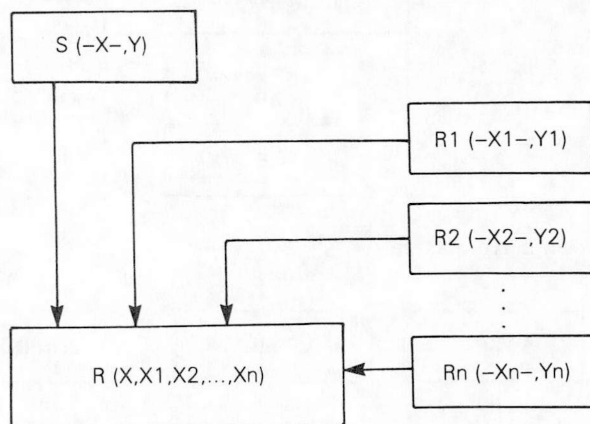
COM	NV	NA	NZ	PC
	V1	A1	Z1	P1
	V1	A1	Z2	P2
	V1	A2	Z1	P3
	V1	A2	Z2	P4

Pero en este caso no existe una DP explícita, sino embebida, entre (NV↠NA | NZ) en la relación COM, y, por tanto, ésta no es descomponible. Es una condición no expresable en el diagrama.

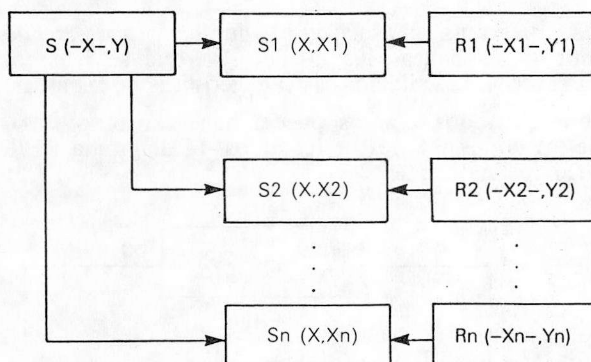
En resumen, cuando al dibujar un diagrama se encuentra una asociación de grado superior a 2, conviene examinar si tiene DPs que permitan descomponerla.

OTROS TIPOS DE DEPENDENCIAS

Sea una asociación R como la representada en el diagrama siguiente:



Supongamos que los valores de un Xi asociados en R a un valor de X son independientes de los de otros Xj asociados al mismo valor de X. O lo que es igual, todas las combinaciones de valores asociados a un X dado son válidas. En este caso existe una DJ(X↠X1 | X2 | ··· | Xn) en la relación R, y por tanto ésta es descomponible en sus proyecciones sobre (X, X1), (X, X2), ..., (X, Xn). Es entonces más correcto, para representar el significado de los datos, dibujar el diagrama como sigue:

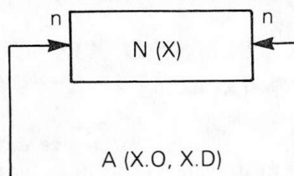


Análogamente, podrían presentarse en R dependencias yuncionales, generalizadas o de otros tipos. Veamos algunos ejemplos de condiciones de integridad no expresables como dependencias.

Ejemplo 1. Representación de un grafo:

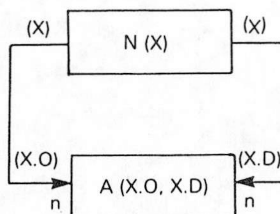
En el siguiente diagrama RE/R representamos la estructura de datos de un grafo plano. Este tipo de estructuras se aplica a todos los problemas que pueden representarse mediante estos grafos (por ejemplo, redes PERT, listas de materiales, organigramas funcionales, etc.).

Sea un grafo bidimensional dirigido, con un arco como máximo, en cada dirección, entre dos nodos. Numeremos todos los nodos. Sea N(X) el conjunto de todos los números de los nodos del grafo. Cada arco es una asociación entre el nodo origen y el nodo destino. Tendremos pues el diagrama C/A siguiente:



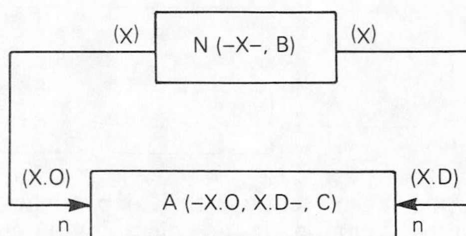
En este diagrama la asociación A entre nodos representa los arcos del grafo. Como es una asociación entre elementos del mismo conjunto (el N(X)), hay que calificar el atributo X para distinguir los dos papeles diferentes que juega en A(X.O: nodo origen; X.D: nodo destino).

Transformemos la asociación explícita A, de cardinalidad (n : n), en dos implícitas funcionales completas. Tendremos así el diagrama RE/R:



(Como en este diagrama las asociaciones implícitas son parciales, hay que indicar explícitamente en las líneas los atributos a que se refieren, como ya se ha comentado cuando se describieron las características de las asociaciones implícitas).

Naturalmente los nodos y arcos pueden tener atributos a su vez. Sean B los atributos de los nodos y C los de los arcos. El diagrama RE/R para representar un grafo quedará finalmente:



El esquema relacional de diseño será:

Relaciones: N(–X–, B), A(–X.O, X.D–, C).

Condiciones: **A**[X.O]⊆**N**[X]; **A**[X.D]⊆**N**[X].

Ejemplo 2. Grafo conexo:

Supongamos ahora que queremos representar un grafo conexo. Es decir, en el que no haya nodos sueltos. Tendremos que todo nodo participa en algún arco. El esquema relacional de diseño será:

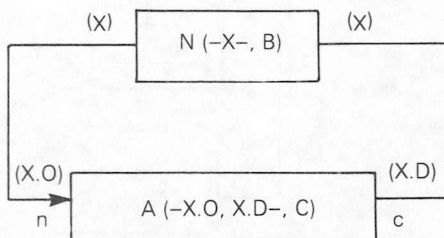
Relaciones: N(–X–, B), A(–X.O, X.D–, C).

Condiciones: **N**[X] = (**A**[X.O] ∪ **A**[X.D]).

Obsérvese que la condición anterior no es expresable ni como una dependencia ni como una condición de integridad de referencia.

Ejemplo 3. Arbol:

Supongamos ahora que queremos representar un árbol. Todo nodo cuelga de otro, excepto el raíz, que no cuelga de ninguno. El diagrama RE/R será:



Como una de las asociaciones implícitas tiene cardinalidad c, su correspondiente atributo, (X.D), tiene que ser clave en A. Por tanto, el esquema de diseño será:

Relaciones: N(–X–, B), A(X.O, –X.D–, C).

Condiciones: A[X.O]⊆N[X]; A[X.D]⊆N[X].

Además de las condiciones de integridad anteriores, deberá cumplirse que en A[X.D] deben aparecer todos los nodos de N[X], menos el raíz (o sea, que A deberá tener una fila menos que N). De nuevo esta condición de integridad no es expresable como uno de los tipos de dependencias estudiados anteriormente ni como integridad de referencia.

Además tiene que cumplirse otra condición de integridad: que no haya bucles en el grafo. Tampoco esta condición es expresable mediante los tipos de condiciones hasta aquí estudiados, y plantea además una situación interesante, pues la detección de bucles implica el uso de operaciones de clausura, es decir, yunciones reiterativas aplicadas un número variable de veces, lo que no es directamente expresable en álgebra relacional.

En resumen, al dibujar el diagrama hay que estar atento a las asociaciones de grado superior a 2, para analizar si pueden contener Dependencias Plurales o de otro tipo que permitan descomponerlas en otras más simples, y hacerlo así si es posible.

USO DE LOS DIAGRAMAS RE/R EN EL DISEÑO DE ESTRUCTURAS IMS

El IMS (Information Management System) es un sistema de Bases de Datos no relacional. Permite diseñar esquemas conceptuales en red con presentación jerárquica. Estos párrafos van dirigidos a los lectores interesados en el IMS.

La construcción de diagramas RE/R nos proporciona un medio de expresión o lenguaje para representar los modelos conceptuales de datos. Puede parecer entonces que estos diagramas serían aplicables tanto a la obtención de esquemas de diseño relacionales, según hemos visto, como a los de otros tipos, por ejemplo esquemas de datos IMS. Sin embargo, no es totalmente así.

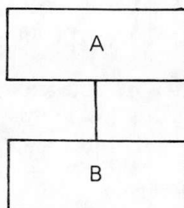
En general, para que así fuera, el lenguaje en el que expresemos el modelo conceptual de datos (el diagrama RE/R en nuestro caso), debería tener al menos la misma potencia expresiva que el del Sistema de Base de Datos que consideremos (en nuestro caso el IMS). Esto no es así si comparamos el lenguaje de un diagrama RE/R con el del modelo conceptual IMS, lo que se debe al diferente enfoque de las construcciones básicas de datos de ambos.

En un diagrama RE/R uno de los elementos básicos, el rectángulo, representa conjuntos de valores, por lo que no pueden éstos repetirse. Sin embargo, en un diagrama conceptual IMS, un rectángulo representa un conjunto de ocurrencias de un determinado tipo de segmento, sin tener en cuenta los valores que éstos contienen, por lo que es posible que estos conjuntos contengan valores repetidos (segmentos sin campo de secuencia o con campo de secuencia múltiple).

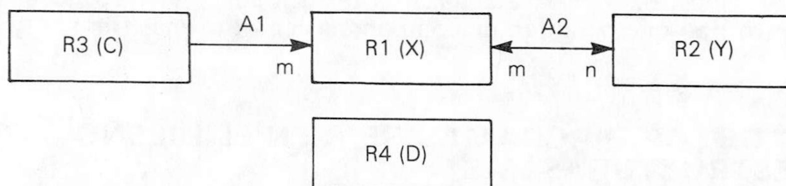
Veamos algún ejemplo para concretar estas ideas.

Ejemplo 1:

Sea el siguiente modelo conceptual IMS:



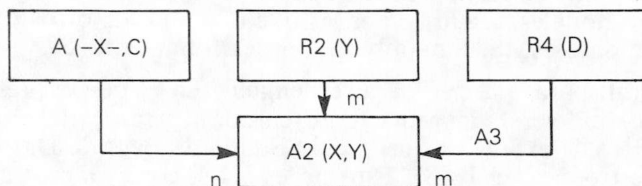
Supongamos que el segmento padre A contiene los campos (X, C) y el hijo B los campos (Y, D), y que los atributos X e Y, posiblemente compuestos, son los respectivos campos de secuencia. Supongamos además que son campos de secuencia únicos, es decir, que los valores de X no se repiten en ninguna ocurrencia del segmento A, y los de Y no se repiten en las ocurrencias de B que cuelguen de un mismo A. Esta situación sí es representable en un diagrama C/A (y por tanto en un diagrama RE/R). El diagrama C/A sería:



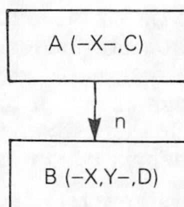
(R1, R2, R3 y R4 son los conjuntos de valores de X, Y, C y D, respectivamente).

Obsérvese que la asociación A2, entre X e Y, es de cardinalidad (m: n), pues un valor determinado de Y puede repetirse bajo distintos valores de X.

El diagrama anterior se transforma en el siguiente diagrama RE/R (en el que aparece una asociación de asociaciones, la A3):



Queda finalmente el diagrama RE/R:



Y, por tanto, el esquema relacional de diseño equivalente a la estructura IMS será:

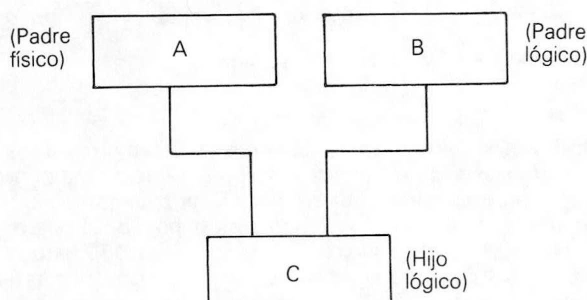
Relaciones: A(–X–, C), B(–X, Y–, D).

Condiciones: **B**[X]⊆**A**[X].

Ni este esquema de diseño ni el diagrama RE/R nos permiten representar la estructura anterior de IMS en el caso de que el campo de secuencia Y sea múltiple (o sea, si Y puede tomar valores repetidos debajo de una ocurrencia dada de A), pues no es posible identificarlas unívocamente por su contenido. Lo mismo ocurriría si el segmento de tipo B no tiene campo de secuencia.

Ejemplo 2:

Veamos cómo se construiría el diagrama RE/R equivalente a una estructura IMS con una relación lógica. Sea esta estructura la siguiente:

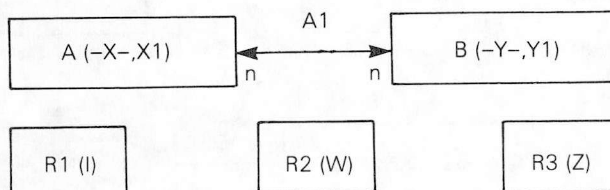


En el diagrama anterior suponemos que A es el padre físico de C, y B es su padre lógico.

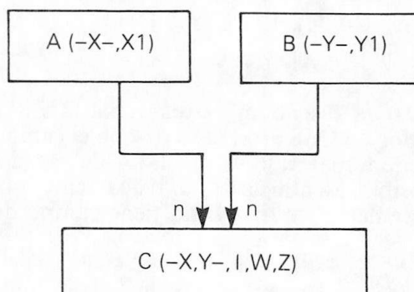
Atributos de A: A(X, X1).
Atributos de B: B(Y, Y1).
Atributos de C: C(I, W, Z).
X: Campo de secuencia único de A.
Y: Campo de secuencia único de B.
I, W, Z: Datos de intersección en C.

Supongamos que las ocurrencias C, vistas desde sus padres, A y B, tienen campos de secuencia únicos. Estos campos, normalmente, serán los de los padres, es decir, entrando por A el campo de secuencia será Y (pues C contiene los campos Y, I, W y Z), y entrando por B será X (C contiene los campos X, I, W y Z).

El diagrama C/A equivalente será:



El diagrama RE/R será:



El esquema relacional de diseño será:

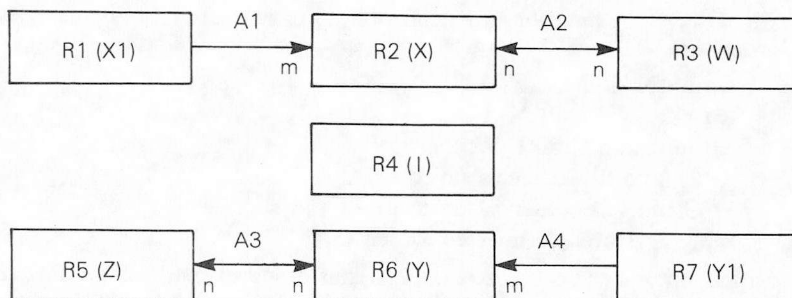
Relaciones: A(–X–, X1), B(–Y–, Y1), C(–X, Y–, I, W, Z).

Condiciones: **C**[X]⊆**A**[X], **C**[Y]⊆**B**[Y].

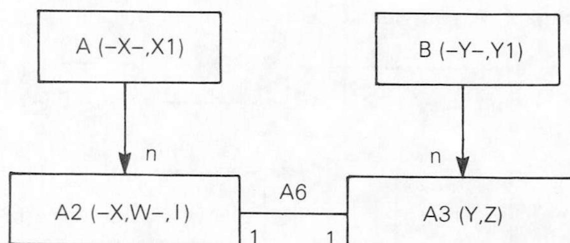
Ejemplo 3:

En general, sin embargo, en IMS, aunque C tenga campos únicos de secuencia, no tienen que ser necesariamente las claves concatenadas de los padres, sino que pueden ser campos cualesquiera de C, por ejemplo, el campo de secuencia entrando por A podría ser W, y entrando por B, Z. Esto permite en IMS que pueda existir más de una ocurrencia del hijo lógico C bajo las mismas ocurrencias de los padres (es decir, más de un C para una misma pareja (A, B) dada).

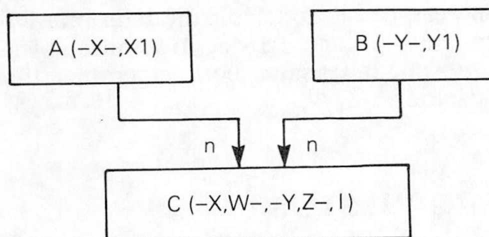
Construyamos el diagrama RE/R para este caso. El diagrama C/A de partida será:



Lo transformamos en:



Lo transformamos finalmente en:



Esquema relacional de diseño:

Relaciones: A(–X–, X1), B(–Y–, Y1), C(–X, W–, —Y, Z–, I).

Condiciones: **C**[X]⊆**A**[X], **C**[Y]⊆**B**[Y].

En resumen, es posible construir un diagrama RE/R equivalente a un modelo conceptual IMS si en éste todos los tipos de segmentos tienen campos «únicos» de secuencia. En caso contrario, no es posible, y, por consiguiente, el diagrama RE/R no será una buena técnica de ayuda para el diseño. En la práctica, sin embargo, es improbable que estos casos se presenten. Si así fuera, podrían evitarse buscando atributos adicionales que puedan emplearse como identificadores (o incluso generándolos en los programas de aplicación).

EL MODELO DE DATOS E/R (DE ENTIDADES/RELACIONES)

El modelo conceptual de datos en un diagrama E/R se construye con los elementos siguientes:

- *Rectángulos con un nombre.* Cada rectángulo representa un conjunto de objetos sobre los que queremos guardar datos homogéneos (por ejemplo: empleados, artículos, etc.).
- *Rombos con un nombre.* Cada rombo representa una asociación entre los elementos de dos o más rectángulos que se unen con líneas al rombo. En los extremos de estas líneas se indica la cardinalidad de la asociación.
- *Ovalos con un nombre.* Cada óvalo representa un conjunto de valores simples, es decir, atributos. Cada óvalo se une con líneas a los rectángulos o rombos para indicar cuáles son los atributos de éstos.

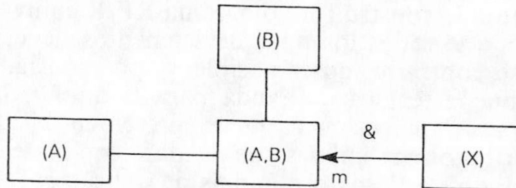
Estos diagramas se llaman diagramas E/R (y también diagramas de Chen). Como vemos no se exige en este modelo de datos que los elementos de un conjunto representado por un rectángulo o un rombo tengan algún atributo identificador o clave. Por ello, las observaciones que decíamos al intentar representar estructuras IMS con diagramas RE/R son también aplicables aquí. Es decir, todo diagrama E/R en el que todos sus rectángulos y rombos tengan clave es representable con un diagrama RE/R equivalente. Además, dada la

similitud entre los elementos básicos de ambos tipos de diagrama, el paso de uno a otro es inmediato. Sin embargo, si hay en el diagrama E/R rectángulos o rombos sin clave, no es posible construir un diagrama RE/R equivalente. En este caso, como decíamos al hablar de IMS, lo aconsejable, si es posible, es añadir nuevos atributos que puedan usarse como identificadores (o generarlos en los programas de aplicación).

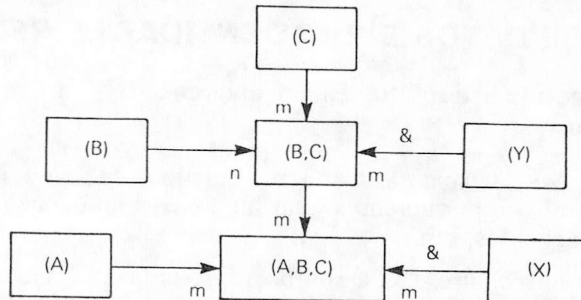
EJERCICIOS PROPUESTOS

En los ejercicios siguientes, para simplificación de los dibujos, dejaremos a veces las asociaciones explícitas sin nombre. En este caso indicaremos que son explícitas adjuntándoles el signo &. También, por la misma razón, se omitirán a veces los nombres de los conjuntos de valores representados en los rectángulos.

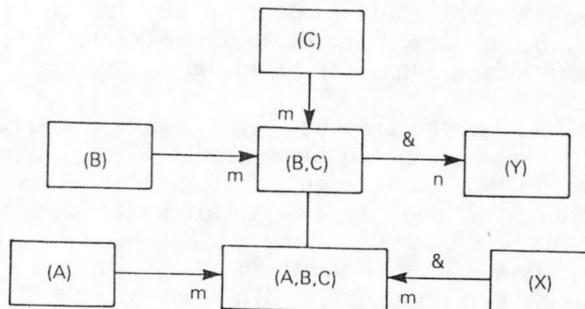
1) *Obtener las relaciones y claves del esquema de diseño a partir del diagrama siguiente:*



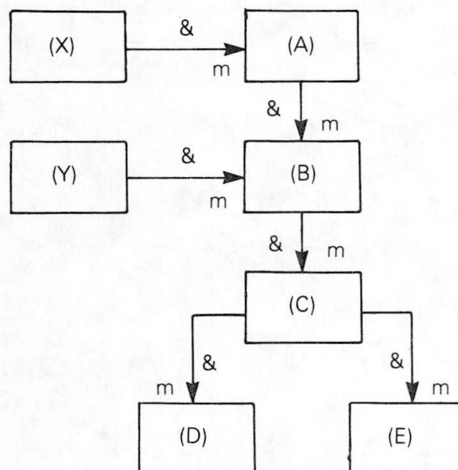
2) *Transformar el diagrama siguiente:*



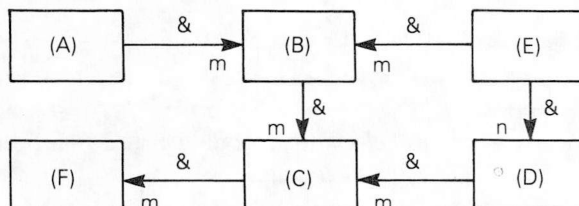
3) *Transformar el diagrama siguiente:*



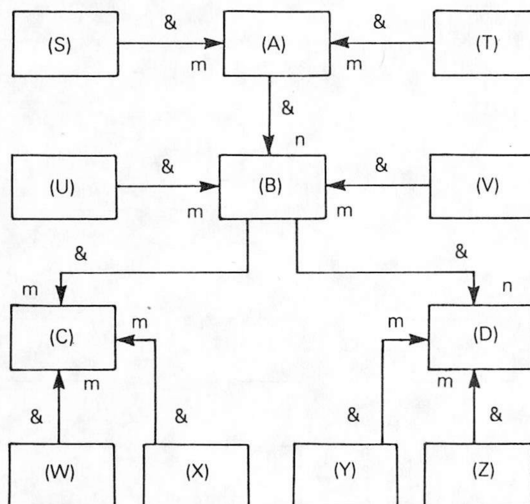
4) *Obtener las relaciones y claves del esquema de diseño a partir del diagrama siguiente:*



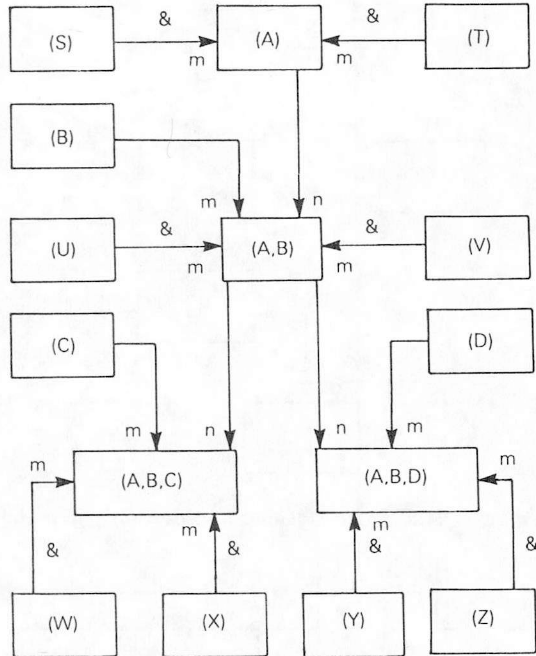
5) *Obtener las relaciones y claves del esquema de diseño a partir del diagrama siguiente:*



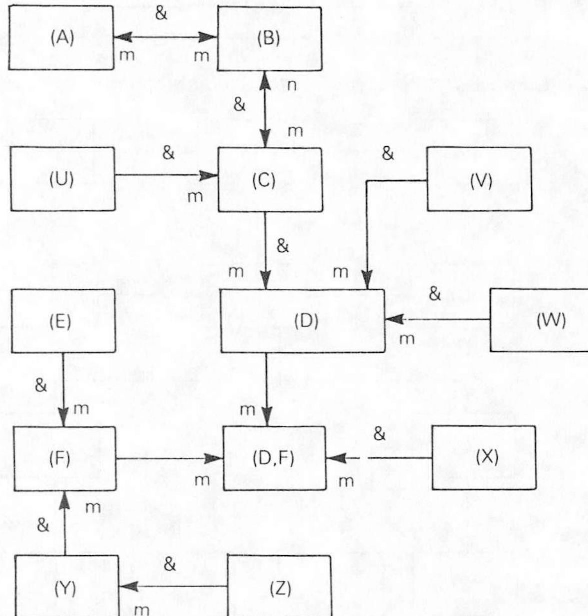
6) *Obtener las entidades definidas por el diagrama siguiente y decir de qué tipo son:*



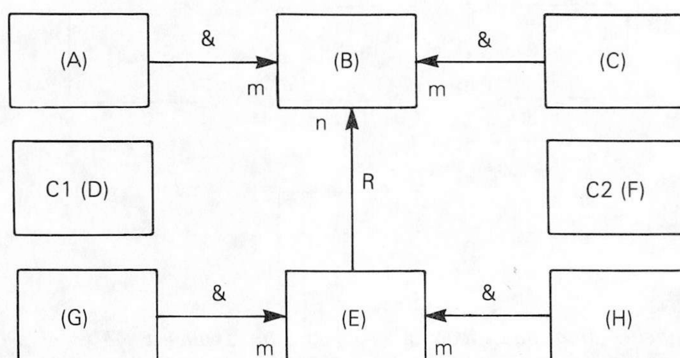
7) *Obtener las entidades definidas por el diagrama siguiente y decir de qué tipo son:*



8) *Obtener las relaciones y claves del esquema de diseño a partir del diagrama siguiente:*

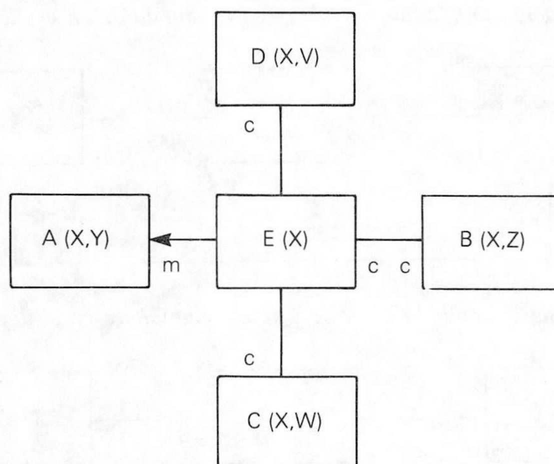


9) Partimos del diagrama siguiente. Además de las asociaciones representadas en él hay otras dos: S y T. La primera entre los elementos de C1 y R, con cardinalidad (1 : m). L a segunda entre los elementos de C2 y T, con cardinalidad (m : m). *Representar S y T en el diagrama, obtener el esquema de diseño y decir de qué tipo son las Entidades obtenidas:*

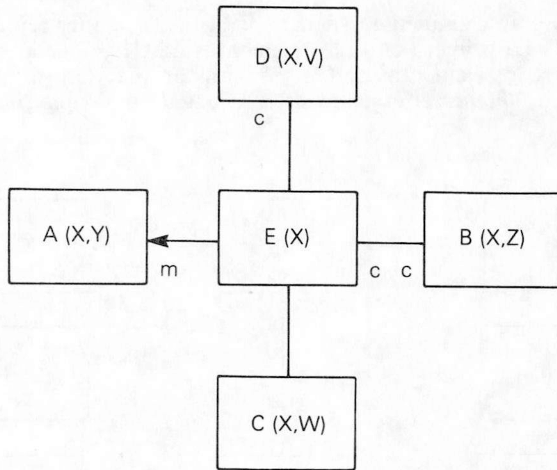


10) Sean dos relaciones, A(X, Y, Z) y B(X, Y, W). La asociación implícita entre A y B, según X, es de cardinalidad (m : 1); según Y, es de cardinalidad (m : c), y según (X, Y), es de cardinalidad (m : c). *Comprobar que estas cardinalidades no son contradictorias y poner un ejemplo de extensiones de ambas relaciones que las cumplan.*

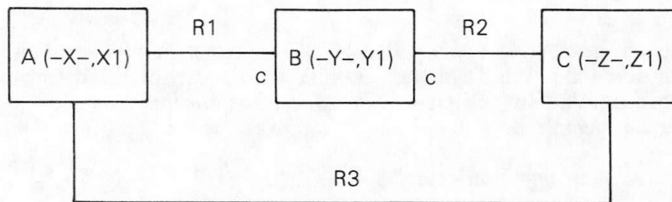
11) *Transformar el diagrama siguiente:*



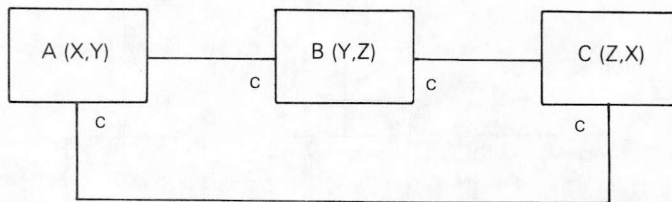
12) *Transformar el diagrama siguiente:*



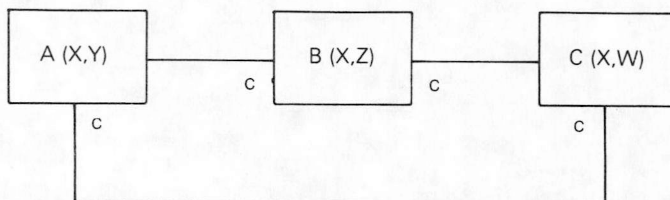
13) *Dado el siguiente diagrama, decir si la asociación explícita R3 es el producto de las asociaciones R1 y R2.*



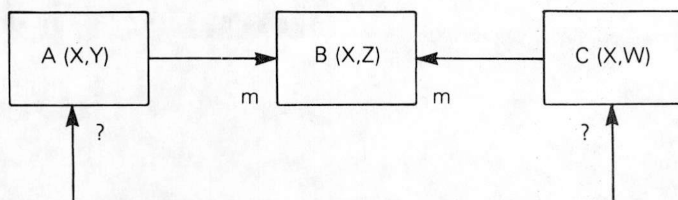
14) *Decir si la asociación implícita entre A y C es redundante en el diagrama siguiente:*



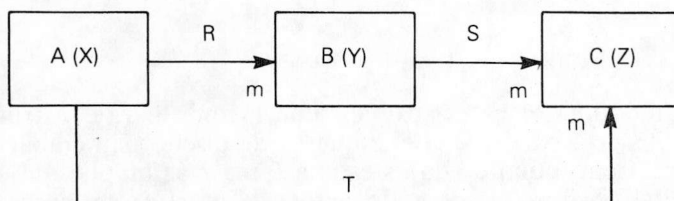
15) *Decir si la asociación implícita entre A y C es redundante en el diagrama siguiente:*



16) *En el diagrama siguiente, hallar las cardinalidades posibles de la asociación implícita entre A y C y decir si es redundante:*



17) *Sea el diagrama siguiente:*



Supongamos que el bucle (R-S-T) se debe a que la asociación T es redundante con R y S. ¿Puede afirmarse también que R es redundante con S y T y que S es redundante con R y T? ¿Cuál de las tres asociaciones, R, S o T, es preferible eliminar en el diagrama?

6 Método y ejemplos de diseño

En el capítulo anterior hemos expuesto un método para construir un diagrama RE/R y obtener a partir de él el esquema de diseño. Sin embargo, el diseño de un Esquema Conceptual de datos es una tarea más amplia que la obtención del diagrama RE/R. En este capítulo vamos a empezar tratando de situar la posición del método propuesto dentro del proceso global de diseño. Seguiremos con un resumen de aquél, y luego expondremos algunos ejemplos de aplicación.

FASES DE DISEÑO

Vamos a considerar en el proceso de diseño las siguientes Fases y actividades:

Fase I

Obtención del Esquema Conceptual de datos. Suele designarse esta fase como Diseño Lógico. Incluye las actividades siguientes:

— I.1). Definir el objetivo del diseño.

Esta labor debe realizarla fundamentalmente la Dirección.

— I.2). Definir los datos a incluir en el Modelo Conceptual.

Aquí hay que decidir qué datos hay que incluir en el modelo, y definir su significado. Para llevar a cabo esta actividad el analista tendrá que obtener información de los usuarios de los datos, frecuentemente a nivel directivo.

La definición precisa del significado de los datos es probablemente la labor más importante y difícil del diseño. Esperamos que algo de esto pueda apreciar el lector cuando examine los ejemplos de este capítulo. Por ello es útil ir construyendo ya a lo largo de esta actividad el diagrama RE/R. De esta forma disponemos de un soporte documentado de las informaciones y decisiones que se vayan tomando. Además de ser una

buena herramienta para afinar con precisión el significado de los datos, también lo es para formalizar y facilitar la comunicación con los usuarios y dirección.

— I.3). Obtener el Esquema Conceptual de datos:

Al final de la actividad anterior, como consecuencia de las definiciones obtenidas en ella y aplicando las reglas de transformación descritas en el capítulo precedente, se obtiene el diagrama RE/R, que representa un primer Esquema Conceptual de datos. Este diagrama se traduce directamente a un esquema de diseño formado por relaciones y condiciones de integridad.

El método de diseño propuesto se aplica plenamente en la realización de esta actividad.

— I.4). Refinar el Esquema de diseño.

El modelo de datos representado por el diagrama RE/R anterior puede necesitar algunas modificaciones o adiciones que se introducirán en las siguientes actividades:

- Añadir a las condiciones de integridad obtenidas directamente del diagrama las que no se hayan podido expresar o utilizar en su construcción.
- Repasar y validar con usuarios el diagrama final obtenido.

  Comprobar que las Entidades obtenidas corresponden a conceptos lógicos y naturales en el mundo real representado por nuestro modelo de datos. Analizar si hay discrepancias. Estas pueden deberse a cardinalidades especificadas incorrectamente.
- Prever si puede haber cambios futuros en la estructura de datos que afecten al diseño.

  Analizar cómo repercutirían en él y la conveniencia de modificarlo ahora para que sean fácilmente asimilables cuando se produzcan.

  Por ejemplo, supongamos que tenemos una asociación de cardinalidad (1 : m) entre Vendedores y Productos que por cambios en la política comercial de la empresa pueda ser en el futuro (m : m). Sería conveniente definirla ya así en el diagrama, lo que podría dar lugar a alguna relación más en nuestro esquema de diseño.
- Incluir la posibilidad de valores nulos.

  Las reglas aplicadas en la obtención del diagrama RE/R producen un diseño en que no hay columnas con valores nulos. Si nuestro Sistema Relacional es capaz de tratar estos valores, podría ser ventajoso considerarlos. Comentaremos con más detalle este punto en otro apartado, más adelante.
- Definir qué claves van a ser Primarias. Definir las reglas de Inserción, Actualización y Borrado para el mantenimiento de las condiciones de Integridad de Referencia.

Fase II

Reajustes al diseño en función de su utilización.

Suele conocerse esta Fase como Diseño Físico.

El diseño obtenido como consecuencia de las actividades anteriores se basa en la estructura e interrelaciones de los datos, sin tener en cuenta qué uso se va a hacer de ellos. Para analizar este punto vamos a clasificar las formas de utilización de los datos en tres tipos:

1) Utilización de los datos por programas de ejecución frecuente, normalmente escritos por programadores profesionales con experiencia.

2) Utilización por programas que se ejecutan un número limitado de veces.

3) Utilización por usuarios finales mediante lenguajes de consultas.

En el primer caso prevalecerán los requerimientos de buen rendimiento (por ejemplo, medido en número de acceso a disco). En el segundo será más importante una productividad alta en programación y pruebas. En el tercero será más importante la facilidad de uso. En caso de tener varios tipos de utilización tendremos que satisfacer simultáneamente requerimientos de uno y otro tipo, lo que puede llevar a criterios de diseño contradictorios, entre los que habrá que buscar un equilibrio.

No incluimos aquí una discusión de cómo repercuten los requerimientos de rendimiento en el diseño físico, pues esto va íntimamente ligado al Sistema Relacional que utilicemos. En general, nos llevará a definir índices o vías de acceso adicionales en disco para las búsquedas más frecuentes. Un cierto grado de redundancia de datos (desnormalización), que siempre debe estar bien controlada y documentada, puede favorecer el rendimiento de operaciones de consulta, en detrimento de las de actualización. Por otra parte, un diseño normalizado favorecerá, en general, a la productividad en desarrollo de programas.

Si la utilización va a ser fundamentalmente para consultas de usuarios finales, podemos facilitarles el uso de los datos mediante acciones como las siguientes:

1) Evitar el abuso de códigos. Así, por ejemplo, en vez de usar códigos de provincias, utilizar sus nombres.

2) Incluir campos precalculados.

3) Incluir vistas con las yunciones de uso más frecuente predefinidas.

Esta lista sólo pretende mostrar algunos ejemplos. En general, conseguimos mayor facilidad de uso a costa de una mayor redundancia de datos.

INCORPORACION DE VALORES NULOS

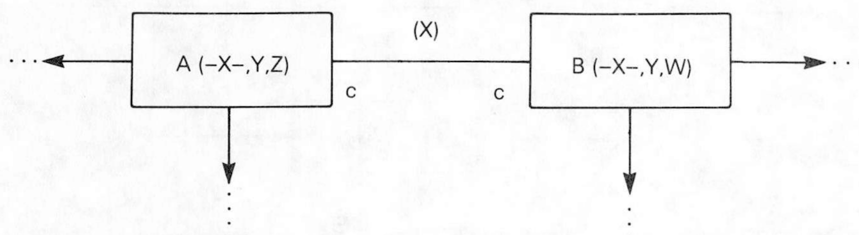
Las reglas de transformación de diagramas que hemos visto y que nos permiten llegar al diagrama RE/R final no tienen en cuenta la posibilidad de que

existan valores nulos. Sin embargo, si el sistema los permite, podríamos disminuir el número de relaciones del esquema de diseño si incluyéramos algunas columnas que admitieran estos valores.

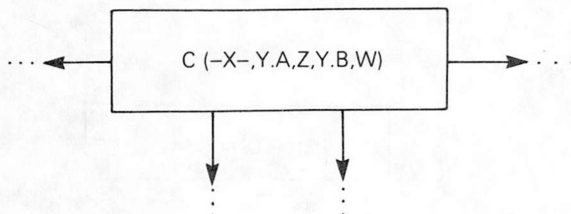
Para conseguir esto vamos a modificar la regla de Agregación de Rectángulos, generalizándola como sigue:

Sean dos rectángulos, A(X, Y, Z) y B(X, Y, W), con atributos comunes X e Y, posiblemente compuestos. Supongamos que X es una clave de A y de B, y que la asociación implícita entre A y B, según X, es de cardinalidad (c : c), o (c : 1), o (1 : 1). Entonces A y B pueden agregarse, substituyéndolos por un solo rectángulo C obtenido como yunción externa entre A y B. Todas las asociaciones en que intervienen A y B se aplican también a C. La clave de C es X.

Veamos el caso gráficamente. Sea el diagrama:



Sea C el resultado de obtener la yunción externa, según X, entre A y B. Entonces el diagrama anterior es equivalente a:



En C todos los atributos, excepto X, pueden tomar valores nulos.

Si la cardinalidad de la asociación entre A y B fuera (1 : c), los atributos de C que podrían tomar valores nulos serían los procedentes de B, es decir, Y.B y W. Si la cardinalidad entre A y B fuera (1 : 1), ningún atributo de C tomaría valores nulos. Este último es el caso que se incluyó en la Regla de Agregación de Rectángulos del capítulo anterior.

Por tanto, una vez obtenido el diagrama RE/R le aplicaremos la nueva Regla de Agregación si consideramos conveniente incluir algunas columnas con la posibilidad de tomar valores nulos. Estos valores pueden ser complicados de entender y manejar para los usuarios finales, por lo que su inclusión en el diseño debe ser discrecional, según los requerimientos de uso previstos para los datos.

6

Ejemplo:

Sean los conjuntos siguientes:

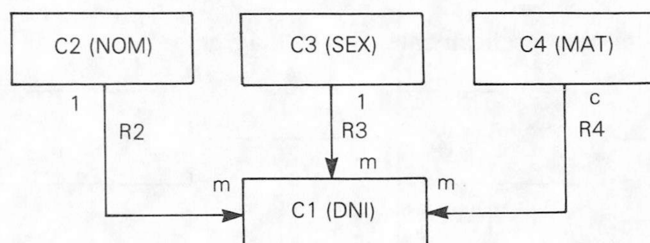
C1(DNI): Conjunto de valores del DNI (Documento Nacional de Identidad) de los empleados de una empresa.

C2(NOM): Conjunto de los nombres de los empleados.

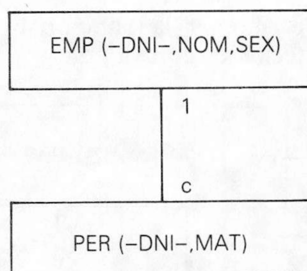
C3(SEX): Sexo de empleado.

C4(MAT): Conjunto de números que representan días de permiso por maternidad concedidos a algunas empleadas.

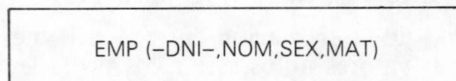
El diagrama C/A de partida será:



Se transforma en el siguiente diagrama RE/R:



Aplicándole la Regla de Agregación generalizada queda:



El diseño final es:

1) Relación: EMP (–DNI–, NOM, SEX, MAT), donde MAT puede tomar valores nulos.

2) Condiciones de integridad: MAT sólo puede tomar valores no nulos si SEX='F'.

6

OBTENCION DEL ESQUEMA RELACIONAL DE DISEÑO

Vamos a resumir aquí el proceso de obtención del diagrama RE/R y del esquema de diseño que se expuso en el capítulo anterior.

1) Representar con rectángulos los conjuntos de valores. Estos pueden ser simples (relaciones unarias) o compuestos (relaciones n-arias).

   Los diseñadores que empiecen a usar el método representarán normalmente conjuntos de valores del grado más sencillo posible. Al ir adquiriendo experiencia representarán conjuntos de grado más complejo, dibujando ya directamente en el diagrama de partida algunas entidades con todos o parte de sus atributos.

2) Representar con rectángulos todas las asociaciones explícitas entre conjuntos, tanto las binarias, funcionales o plurales como las n-arias.

3) Representar con rectángulos las asociaciones explícitas entre asociaciones.

4) Representar con líneas entre los rectángulos todas las asociaciones. Todas ellas serán implícitas. Representar la cardinalidad de las asociaciones sobre las líneas respectivas.

5) Comprobar que no se han definido asociaciones explícitas parciales.

6) Aplicar la regla de Definición de Claves a las asociaciones implícitas (m : c), (m : 1), (1 : c) o (1 : 1).

7) Comprobar si alguna de las relaciones n-arias representadas en los rectángulos no está normalizada. Esto será así si hay dependencias que no sean consecuencia de las claves. Descomponer estas relaciones si es posible.

8) Comprobar si alguna relación es producto de otras. En este caso suprimirla si es posible.

   En esta etapa del proceso hemos llegado ya a obtener un diagrama RE/R inicial, válido para obtener un esquema de diseño, aunque probablemente demasiado prolijo. Conviene transformarlo para simplificarlo. Este es el objetivo de los siguientes pasos.

9) Aplicar reiterativamente mientras sea posible las Reglas de Agregación de Rectángulos (a las asociaciones (1 : 1)) y de Supresión de Rectángulos (en las asociaciones (m : 1)).

   Al aplicar estas reglas el número de rectángulos disminuye o se mantiene igual, por tanto, es un proceso finito.

10) Cuando ya no se pueda transformar más el diagrama RE/R aplicando reglas, habremos llegado a su forma definitiva, de la que obtendremos el esquema de diseño.

El Esquema Relacional de diseño constará de los siguientes apartados:

1) *Tablas.*
Descripción de las Tablas, sus atributos y sus claves primarias.

2) *Condiciones de Integridad deducibles del diagrama.*
Descripción de condiciones de integridad deducibles de las cardinalidades. Descripción de claves adicionales deducibles también de las cardinalidades.

Estas descripciones tienen como objetivo fundamental dejar documentados con precisión el significado e interrelaciones de los datos.

Además, algunas de estas condiciones de integridad se incorporarán a los programas de actualización o al sistema para forzar a que se cumplan, protegiendo así la coherencia de los datos. Es aconsejable hacerlo así para las condiciones de Integridad de Referencia al menos (Reglas de Inserción, Actualización y Borrado, con las opciones de Rechazo, Propagación o Anulación, que se comentaron en el apartado «Modelo Relacional de Datos»). Para otras condiciones menos importantes puede ser más conveniente no forzar su cumplimiento, pues podría introducir demasiada rigidez en el sistema. La experiencia y buen juicio del diseñador decidirán en cada caso. Así, por ejemplo, en el último apartado de este capítulo (B. de D. Universitaria) hay una condición que dice que todo alumno debe pertenecer al menos a un grupo. Esto obliga a asignar un grupo a cada alumno en el momento en que éste se matricule, lo que puede ser administrativamente complicado. Es preferible permitir temporalmente, hasta el comienzo del curso, la existencia de alumnos matriculados sin asignar a ningún grupo.

3) *Condiciones adicionales no deducibles del diagrama.*
Descripción de condiciones de integridad y claves adicionales no deducibles del diagrama. A lo largo de este capítulo veremos varios ejemplos.

Las observaciones sobre el objetivo y uso de las condiciones de integridad que hacíamos en el párrafo anterior son también válidas aquí.

NORMALIZACION DEL DISEÑO

En la mayor parte de los casos la obtención del diagrama RE/R nos conducirá de forma directa a un diseño normalizado FNBC. Puede haber casos, sin embargo, en que para poder expresar todas las Dependencias Funcionales existentes haya que distorsionar la forma espontánea de obtener el diagrama, con lo cual también se podría llegar probablemente a un diseño normalizado, pero a costa de complicar el proceso introduciendo asociaciones o rectángulos un tanto artificiosos o de significado no evidente. Por ello, en principio es preferible no preocuparse de esto, dibujando el diagrama de la forma más directa y natural posible, y repasando al final el diseño producido para incluir en él las Dependencias no tenidas en cuenta, normalizándolo o añadiéndole condiciones de integridad adicionales.

Ejemplo:

Sean los conjuntos siguientes:

C1 (DNI): Conjunto de los DNI de los clientes de un Banco.

C2 (CTA): Conjunto de los números de cuenta del Banco.

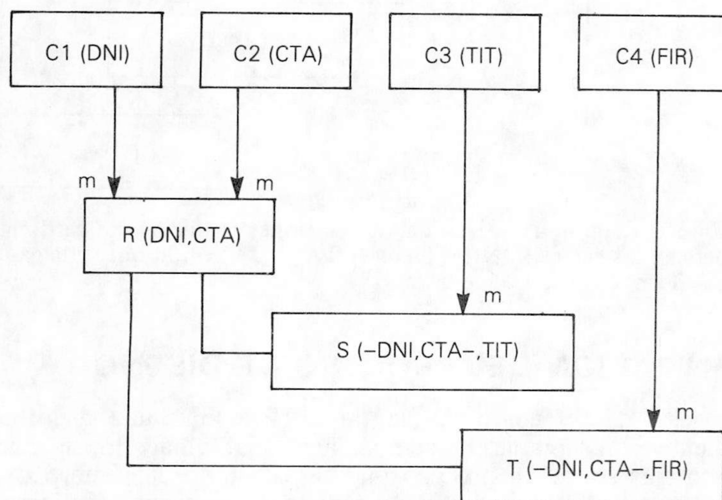
C3 (TIT): Conjunto de números que indican el orden de un Titular dentro de una cuenta.

C4 (FIR): Código que indica si un Titular de una cuenta puede firmar talones de ésta (puede tomar dos valores: 'S' o 'N').

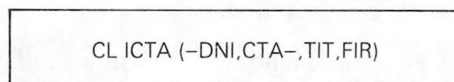
R: Asociación entre los clientes y las cuentas de las que son titulares.

Una cuenta puede tener varios titulares. Un cliente puede ser titular de varias cuentas. Dentro de una cuenta, los clientes que son sus titulares están numerados consecutivamente, siendo éste el número de orden de titular dentro de la cuenta.

Tendremos el siguiente diagrama:



Se transforma en:

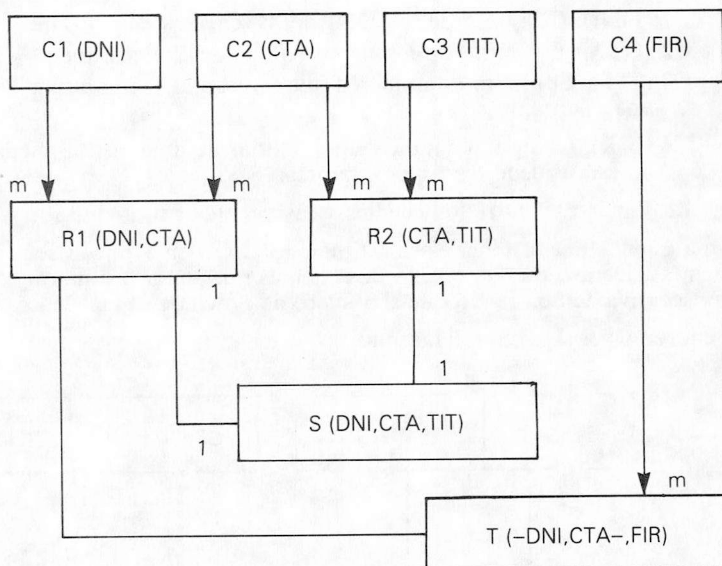


Como dentro de una cuenta no se repiten los números de orden de los Titulares, en esta relación deberá ser clave (CTA, TIT). Es decir, que se cumple que (CTA, TIT→DNI) y (CTA, TIT→FIR). Sin embargo, la única clave obtenida en el diagrama es (DNI, CTA). Para corregir esta situación añadimos la nueva clave al diseño, quedando éste:

Relación: CLICTA (DNI, CTA, TIT, FIR).

Claves: (DNI, CTA) y (CTA, TIT).

Para que el diagrama de partida produzca ambas claves podríamos haberlo dibujado así:



En este diagrama, S tiene dos asociaciones implícitas de cardinalidad (1 : 1), que definen las claves (DNI, CTA) y (CTA, TIT), con lo que se llega al final al mismo diseño normalizado que antes.

MECANIZACION DEL PROCESO DE DISEÑO

El proceso de obtención del diagrama RE/R final, una vez dibujado el inicial, es fácilmente programable. Es además aconsejable hacerlo así, pues proporciona varias ventajas sobre un proceso manual. Cuando el número de rectángulos y asociaciones es elevado, como suele ocurrir en los casos prácticos, la transformación a mano del diagrama, haciendo y rehaciendo el dibujo varias veces, puede ser laboriosa y propensa a errores.

El uso de herramientas automatizadas nos da otras ventajas adicionales:

a) Obliga a una sistemática homogénea independientemente de la persona que realice el diseño.

b) Facilita la obtención de informes para documentación.

c) Facilita la comunicación entre las personas involucradas en el proceso (diseñador, dirección y usuarios) y entre las que lo van a utilizar (analistas, programadores y usuarios finales).

d) Permite analizar fácilmente varias alternativas de diseño para evaluar la más conveniente. Por ejemplo, evaluar el impacto de futuros cambios en las estructuras de datos.

e) Facilita el trabajo más tedioso del proceso, liberando al diseñador para labores más creativas.

A modo de ejemplo de la ayuda que un programa sencillo de este tipo puede ofrecer, se incluyen en los Apéndices los resultados obtenidos al aplicar un programa a uno de los ejemplos que se presentan a continuación en este capítulo.

EJEMPLOS

A continuación se exponen algunos casos prácticos de diseño, muy simplificados en cuanto al número de atributos para que destaquen los aspectos interesantes sin perdernos en detalles demasiado prolijos.

En los diagramas representaremos las asociaciones implícitas sin nombre, como hemos venido haciendo hasta aquí. Tampoco en las explícitas, a veces, pondremos nombre para simplificar el dibujo del diagrama. En este caso indicaremos que se trata de una asociación explícita poniéndole el signo &. Esto lo haremos normalmente cuando un rectángulo sólo participe en una asociación de cardinalidad (m : 1), pues no merece la pena ponerle un nombre que va a desaparecer al aplicar la regla de Supresión de Rectángulos.

Ejemplo 1. Base de Datos de Fábricas y Países:

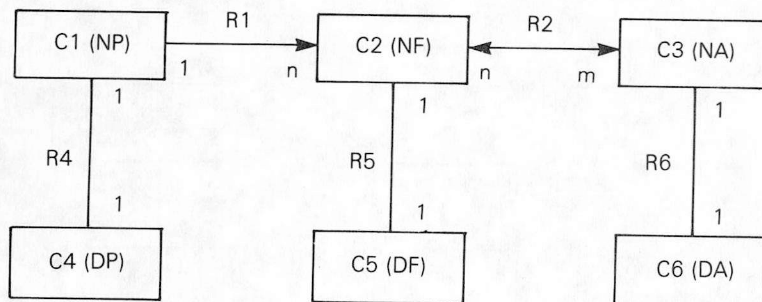
Conjuntos de valores de partida:

C1 (NP): Conjunto de números identificadores de países.
C2 (NF): Conjunto de números identificadores de fábricas.
C3 (NA): Conjunto de números identificadores de artículos.
C4 (DP): Conjunto de descripciones de países.
C5 (DF): Conjunto de descripciones de fábricas.
C6 (DA): Conjunto de descripciones de artículos.
C7 (PR): Conjunto de precios de coste de los artículos producidos por todas las fábricas.

Asociaciones:

R1: Fábrica reside en país.
R2: Fábrica produce artículos.
R3: Precio asociado a un artículo producido por una fábrica.

Diagrama C/A:

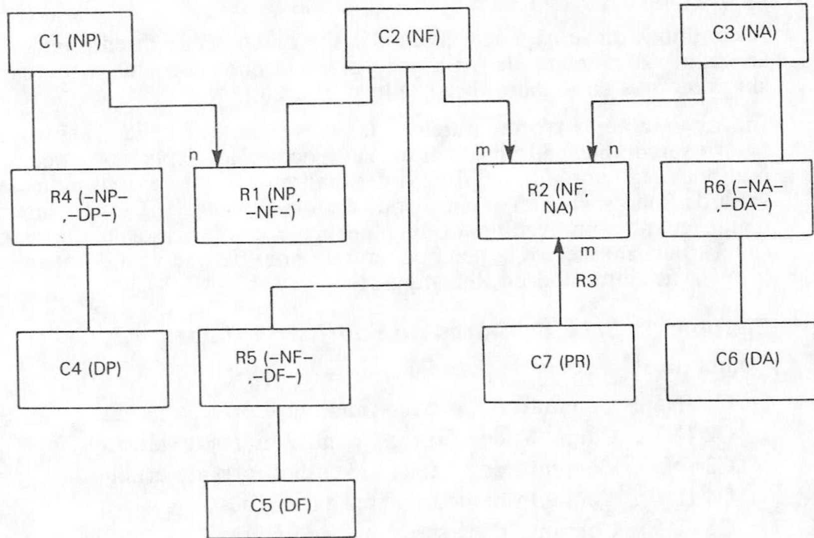


Condición adicional:

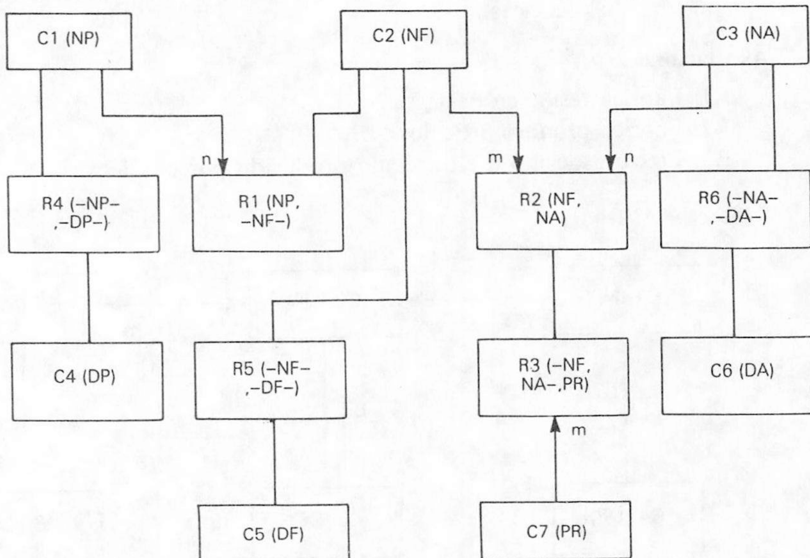
Dentro de un país, un determinado artículo no puede ser producido más que en una fábrica.

Esta condición no se ha representado en el diagrama.

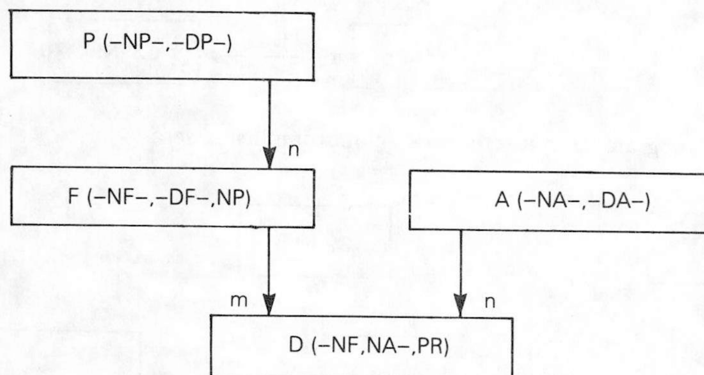
Transformemos todas las anteriores asociaciones en implícitas y representemos también el precio de coste (R3):



Transformemos la asociación R3 en implícita:



Agregando y suprimiendo rectángulos queda:



Esquema de diseño:

1) Tablas:

Países: P(–NP–, –DP–).
Fábricas: F(–NF–, –DF–, NP).
Artículos: A(–NA–, –DA–).
Fábricas que producen Artículos y a qué coste: D(–NF, NA–, PR).
Claves primarias: NP (en P), NF (en F), NA (en A).

2) Condiciones deducibles del diagrama:

$\mathbf{F}[NP] \subseteq \mathbf{P}[NP]$.
$\mathbf{D}[NF] = \mathbf{F}[NF]$.
$D[NA] \subseteq \mathbf{A}[NA]$.

La segunda condición anterior es más fuerte que una de Integridad de Referencia. Las otras dos son condiciones de Integridad de Referencia. Sus claves ajenas son F.NP y D.NA respectivamente.

3) Condición no expresada en el diagrama:

En (F * D), (NP, NA) es clave.

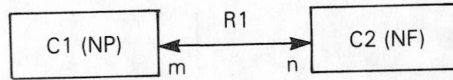
Por tanto, si $E = \mathbf{P}(NF, NA, NP, PR)$ (F * D), se verifica en E que: $NF \rightarrow NP$; $(NA, NP) \rightarrow NF$; $(NA, NP) \rightarrow PR$, y además las claves son (NP, NA) y (NF, NA). Por lo tanto, E no está normalizada FN3.

Ejemplo 2. Otra vez la Base de Datos de Fábricas y Países

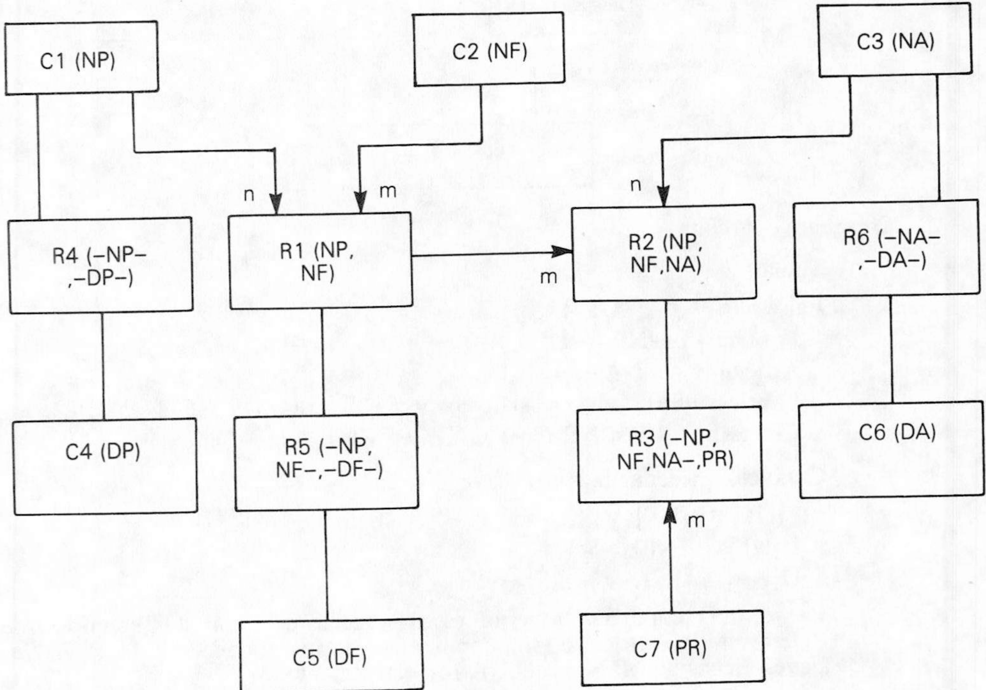
Supongamos el mismo ejemplo anterior, pero con la hipótesis adicional de que las fábricas están identificadas por números dentro de cada país. Por ejemplo:

País	*Fábricas que tiene*
1	1, 2, 3
2	1, 2
3	1, 2, 3, 4

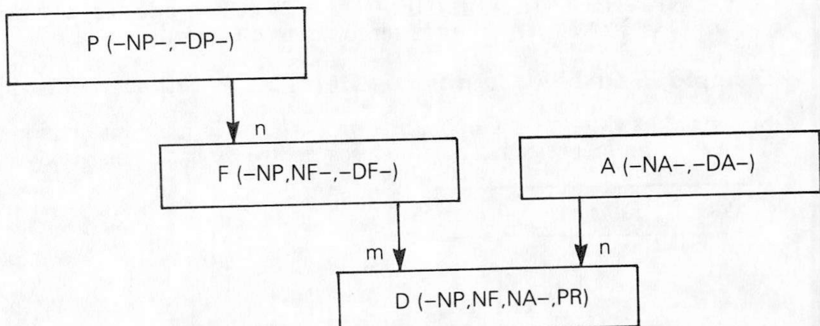
Tendremos entonces que la asociación R1 es ahora (m : n):



El diagrama RE/R, con asociaciones implícitas será:



Agregando y suprimiendo rectángulos:



Este diagrama es como el del ejemplo 1, pero con la diferencia de que el identificador de la Entidad «Fábrica» se compone de NP (identificador de país) y NF (identificador de fábrica dentro de país).

Como (NP, NA) debe ser una clave en D, el esquema será el siguiente:

Esquema de diseño

1) Tablas:

P(–NP–, –DP–). Clave primaria: NP.
F(–NP, NF–, –DF–). Clave primaria: (NP, NF).
A(–NA–, –DA–). Clave primaria: NA.
D(–NP, NA–, NF, PR).

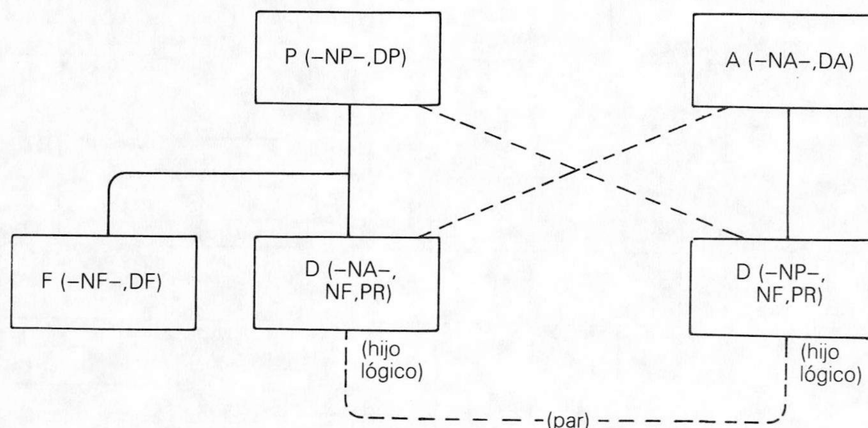
Obsérvese que en D no se cumple NF→NP, a diferencia de lo que ocurre en E en el ejemplo 1, donde sí se cumple. El presente diseño está normalizado.

2) Condiciones:

F[NP]⊆**P**[NP]; **D**[NP, NF]=**F**[NP, NF]; **D**[NA]⊆**A**[NA].

Este diseño es mejor que el obtenido en el ejemplo 1. En general, es aconsejable que los identificadores de entidades dependientes sean compuestos (en nuestro ejemplo, el identificador de Fábrica).

El correspondiente diseño en IMS sería:



Ejemplo 3. Parte de la B. de D. de una Empresa Industrial

Consideremos los conjuntos siguientes:

C1 (ND): Identificadores de Departamentos.
C2 (NE): Identificadores de Empleados.
C3 (NY): Identificadores de Proyectos.

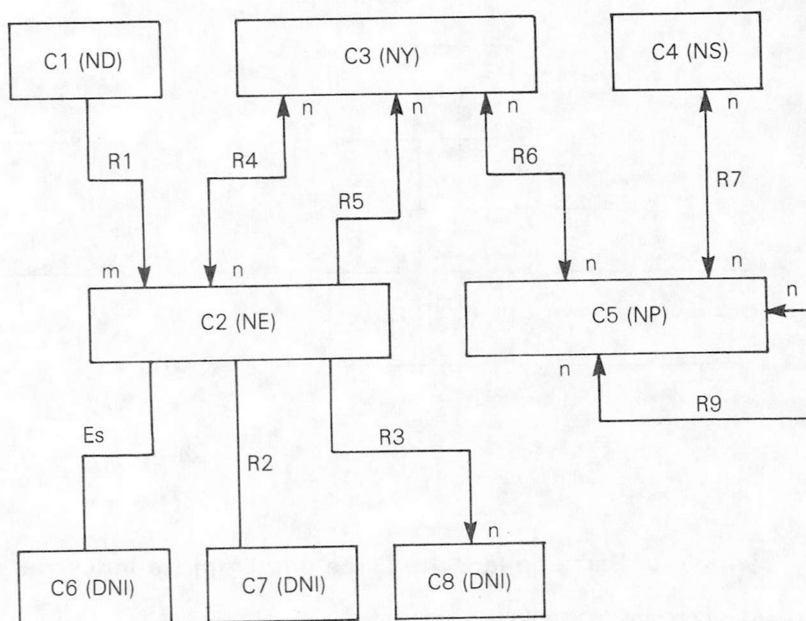
C4 (NS): Identificadores de Suministradores.
C5 (NP): Identificadores de Productos.
C6 (DNI): DNIs de los Empleados.
C7 (DNI): DNIs de los cónyuges de los Empleados.
C8 (DNI): DNIs de los hijos de los Empleados.

Asociaciones:

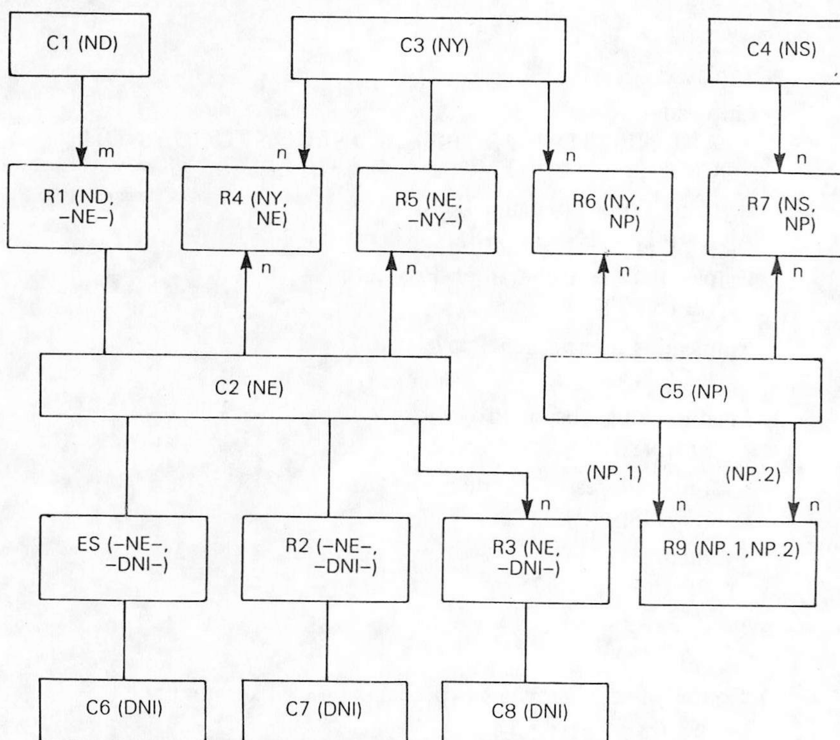
R1: Empleado trabaja en Departamento.
R2: Empleado casado con cónyuge.
R3: Empleado es padre de hijos.
R4: Empleado trabaja para Proyecto.
R5: Empleado dirige Proyecto.
R6: Proyecto necesita Producto.
R7: Suministrador suministra Producto.
R8: Suministrador suministra para Proyecto.
R9: Producto se compone de Producto.

Supongamos que R8 es consecuencia de R6 y R7. La eliminamos por tanto.

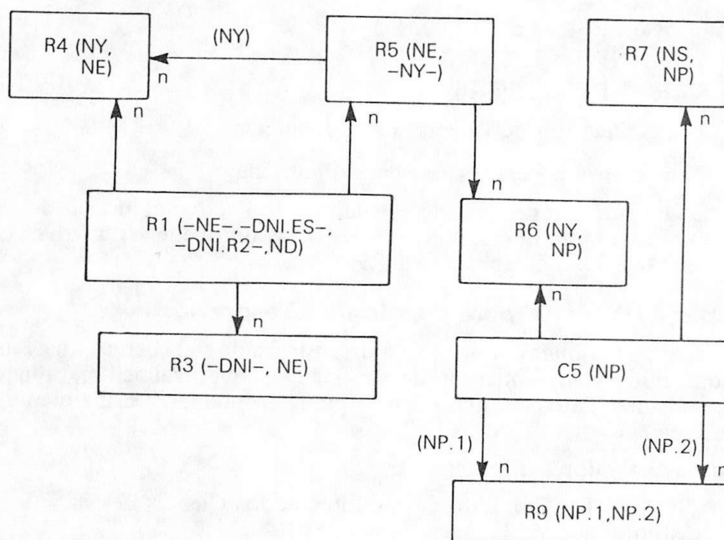
Diagrama C/A inicial:



Transformemos todas las asociaciones en implícitas. Obtenemos así el diagrama RE/R de partida:



Agregando y suprimiendo rectángulos:



Esquema de diseño

1) Tablas:

Empleados:
R1 (–NE–, –DNI.PROPIO–, –DNI.CONYUGE–, ND).
Clave primaria: NE.

Hijos de Empleados con DNI:
R3 (–DNI.HIJO–, NE).

Empleados que trabajan en Proyectos:
R4 (NY, NE).

Empleados que dirigen Proyectos:
R5 (–NY–, NE).

Productos usados en Proyectos:
R6 (NY, NP).

Suministradores de Productos:
R7 (NS, NP).

Productos (NP.1) que se componen de otros (NP.2):
R9 (NP.1, NP.2).

Productos:
C5 (NP).

2) Condiciones adicionales expresadas en el diagrama:

R4[NE] $\subseteq$ **R1**[NE].
R5[NE] $\subseteq$ **R1**[NE].
R3[NE] $\subseteq$ **R1**[NE].
R4[NY] $\subseteq$ **R5**[NY].
R6[NY] $\subseteq$ **R5**[NY].
R6[NP] $\subseteq$ **C5**[NP].
R9[NP.1] $\subseteq$ **C5**[NP].
R9[NP.2] $\subseteq$ **C5**[NP]

Todas ellas son de Integridad de Referencia.

3) Condiciones no expresadas en el diagrama:

El grafo que representa la estructura de Productos no puede tener bucles, pues una pieza que se compone de otras no puede ser a su vez componente de éstas. Esto es aplicable a R9.

Ejemplo 4. B. de D. para Gestión de Tesorería:

Un Banco desea poner consultable a disposición de sus clientes una Base de Datos con los saldos y movimientos de sus cuentas. Naturalmente, cada cliente sólo podrá consultar datos de unas cuentas predeterminadas, normalmente aquellas de las que sea titular.

Conjuntos de valores a representar:

A1(NOMBRE): Conjunto de nombres de los clientes del Banco. Dos clientes distintos pueden llamarse igual.

A2(TIPOCTA): Conjunto de valores que representan distintos tipos de cuentas. Por ejemplo: 1 = Cuentas corrientes, 2 = Libretas de ahorro, 3 = Imposiciones a plazo fijo, etc.

CLIENTE(#CL): Conjunto de números identificadores de los clientes. Suponemos que el Banco ha asignado uno de estos identificadores a cada uno de sus clientes.

CUENTA(#CTA): Conjunto de números de cuenta de los clientes.

A3(#TIT): Conjunto de los números de orden de los titulares dentro de las cuentas. Se supone que dentro de cada cuenta los titulares están numerados.

A4(FECHA): Conjunto de las fechas en las que vamos a tener información.

A5(SALDO): Conjunto de los saldos de las cuentas en cada fecha.

A6(#MOV): Conjunto de los números identificadores de los movimientos de saldo. Se supone que cada movimiento está numerado dentro de cada cuenta y fecha.

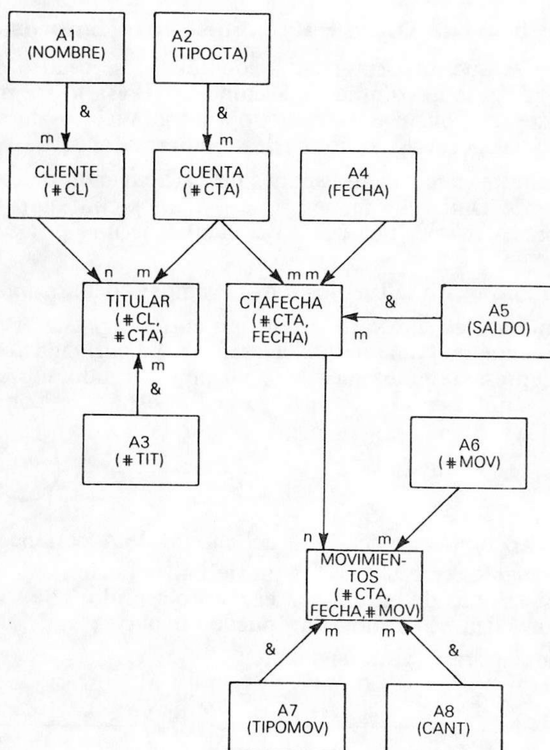
A7(TIPOMOV): Código de tipo de movimiento (ingresos o reintegros).

A8(CANT): Conjunto de las cantidades en pesetas de los movimientos.

Entre clientes y cuentas hay una asociación que llamaremos TITULAR que nos asocia cada cuenta con los clientes que son sus titulares.

La asociación CTAFECHA nos asocia cada cuenta con una fecha, y la asociación MOVIMIENTOS nos asocia cada cuenta y fecha con sus movimientos.

El diagrama RE/R de partida es el siguiente:



Se obtiene el siguiente esquema.

Esquema de diseño

1) Tablas:

CLIENTES (–#CL–, NOMBRE).
CUENTAS (–#CTA–, TIPOCTA).
TITULARES (–#CL, #CTA–, #TIT).
SALDOS (–#CTA, FECHA–, SALDO).
MOVIMIENTOS (–#CTA, FECHA, #MOV–, TIPOMOV, CANT).

2) Condiciones expresadas en el diagrama:

Todo identificador de cliente en TITULARES debe existir en CLIENTES.

Todo identificador de cuenta en TITULARES debe existir en CUENTAS, y viceversa.

Todo identificador de cuenta en SALDOS debe existir en CUENTAS, y viceversa.

Toda pareja de valores de (#CTA, FECHA) en MOVIMIENTOS debe existir en SALDOS.

3) Condición no expresada en el diagrama:

En TITULARES, la pareja de atributos (#CTA, #TIT) es una clave alternativa.

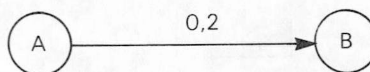
Ejemplo 5. B. de D. de Relaciones entre Empresas

Supongamos que una Empresa puede adueñarse total o parcialmente del capital de otra mediante la compra de acciones en Bolsa u operaciones de otro tipo, sin ninguna restricción. De este modo podría incluso darse el caso de que una empresa filial poseyera acciones de su empresa matriz.

Deseamos almacenar las relaciones de participación de capital entre empresas en una Base de Datos, de manera que podamos contestar a la pregunta de, dadas dos empresas A y B, qué parte del capital de una posee la otra, si es que posee alguna.

Antes de entrar a analizar los datos, vamos a definir algo más el problema.

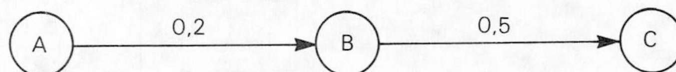
Representemos en un grafo la relación entre empresas. Para ello cada empresa la representamos con un nodo del grafo, y la participación de capital de una en otra la representaremos mediante un arco dirigido sobre el que se indicará qué parte del capital de una es poseído por la otra. Así, por ejemplo:



Este gráfico indica que el 20 % del capital de A pertenece a B.

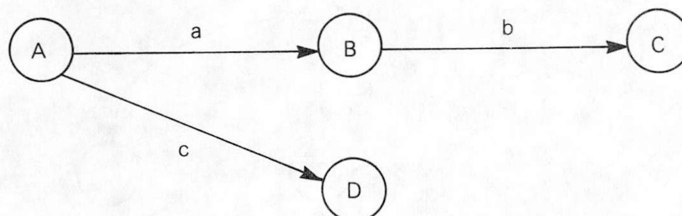
Evidentemente, en estos grafos puede haber bucles. Además, entre dos nodos no puede haber más de un arco en el mismo sentido, y la suma de los valores de los arcos que salen de un nodo no puede ser mayor que 1.

Tomemos ahora el grafo siguiente:



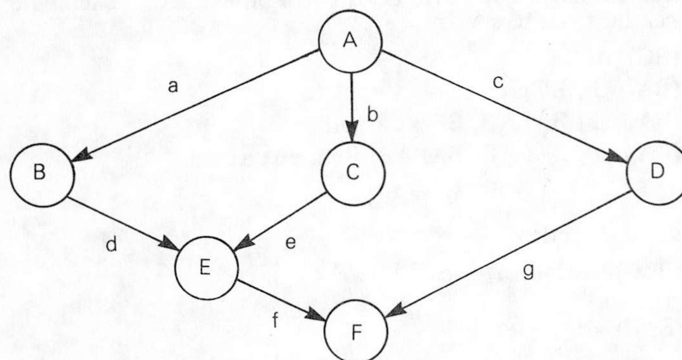
Vemos en él que B posee el 20 % del capital de A, y C el 50 % del de B. Esto significa que, indirectamente, C posee el 10 % de A. Esto lo representaremos con esta notación: (A : C) = 0,1.

Veamos otro ejemplo:



Si en este grafo todo el capital de A está en manos de B y D, deberá ser (a + c) = 1. La participación de C en A será: (A : C) = (a · b).

En general, para hallar qué parte del capital de una empresa está en manos de otra hay que recorrer todos los caminos entre los nodos que las representan y sumar los productos de los arcos de cada camino. Veamos otro ejemplo:



Hay tres caminos de A a F:

(AB, BE, EF): a d f.

(AC, CE, EF): b e f.

(AD, DF): c g.

Por tanto, la parte de A que es de F será:

(A : F) = (a d f + b e f + c g).

Evidentemente, si en el grafo anterior se ha representado todo el capital de A, B, C, D y E, la empresa A pertenece totalmente a F. En efecto, en este caso tendremos que:

a + b + c = 1.

d = 1.

e = 1.

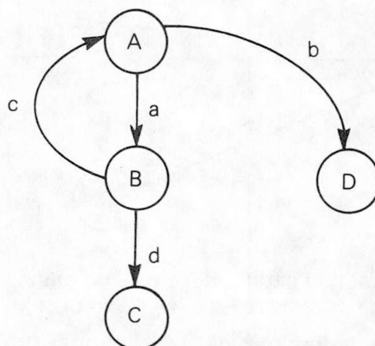
g = 1.

f = 1.

Por tanto:

$$(A:F)=(adf+bef+cg)=1$$

Veamos ahora un ejemplo con bucle:



Queremos hallar qué parte de B corresponde a C. Tenemos que recorrer todos los caminos posibles:

(BC): d.
(BA, AB, BC): c a d.
(BA, AB, BA, AB, BC): c a c a d.
(BA, AB, BA, AB, BA, AB, BC): c a c a c a d.
...

La suma de todos ellos:

$$(B:C)=d\left(1+(ca)+(ca)^2+(ca)^3+\cdots\right).$$

$$(B:C)=d\frac{1}{1-ca}$$

Si el grafo anterior representa todo el capital de A y B, sus únicos dueños serán C y D.

En efecto, en este caso:

$$a+b=1$$
$$c+d=1$$

Tendremos:

$$(A:C)+(A:D)=ad(1+ac+(ac)^2+\cdots)+b(1+ac+(ac)^2+\cdots)=$$
$$=ad\cdot\frac{1}{1-ac}+b\cdot\frac{1}{1-ac}=\frac{ad+b}{1-ac}=\frac{a(1-c)+(1-a)}{1-ac}=1$$

Volvamos ahora a nuestro problema inicial. Nuestra Base de Datos deberá representar el grafo. Para ello tendrá que representar el conjunto de nodos (empresas) y el de arcos (es decir, participaciones directas). Cada arco es una asociación entre dos nodos.

Para hallar la participación de una empresa en otra construiremos un algoritmo que recorra todos los caminos entre sus nodos. Si no hay bucles será finito. Si hay bucles, tendrá infinitas iteraciones. Lo terminaremos cuando hayamos obtenido una aproximación que estimemos suficiente. Es decir, cuando el resultado de la iteración (n+1) difiera en menos de una cantidad predeterminada del obtenido en la iteración (n).

Conjuntos de valores que deseamos representar en nuestra Base de Datos:

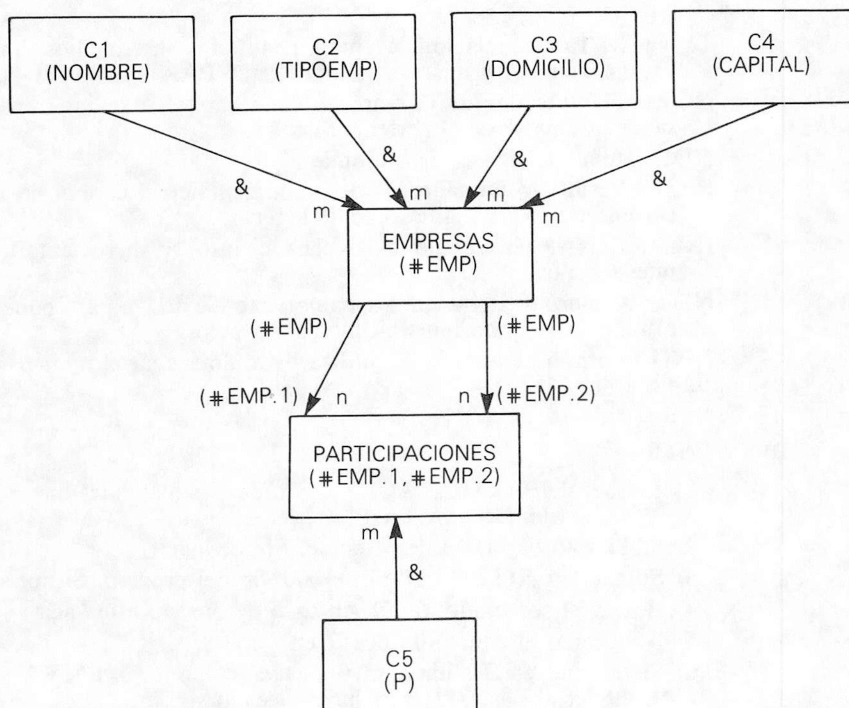
C1(NOMBRE): Conjunto de nombres de las Empresas.
C2(TIPOEMP): Conjunto de códigos que indican tipo de empresa.
C3(DOMICILIO): Conjunto de domicilios de las empresas.
C4(CAPITAL): Conjunto de cantidades en pesetas que representan los capitales de las empresas.
EMPRESAS(#EMP): Conjunto de números identificadores de las empresas.
C5(P): Conjunto de cantidades que representan participaciones directas de capital de unas empresas en otras, en porcentajes.

Los arcos del grafo los representaremos mediante la asociación:

PARTICIPACIONES (#EMP.1, #EMP.2)

donde #EMP.1 es el nodo origen del arco y #EMP.2 el destino.

El diagrama RE/R de partida es:



El esquema resultante es el siguiente:

Esquema de diseño

1) Tablas:

EMPRESAS (–#EMP–, NOMBRE, TIPOEMP, DOMICILIO, CAPITAL).
PARTICIPACIONES (–#EMP.1, #EMP.2–, P).

2) Condiciones expresadas en el diagrama:

Los valores de #EMP.1 y #EMP.2 en la tabla PARTICIPACIONES deben existir en la columna #EMP de la tabla EMPRESAS (Integridad de Referencia).

3) Condiciones no expresadas en el diagrama:

La suma de los valores de la columna P de todos los arcos que salen de un nodo no puede superar a 100.

Con estas tablas podemos utilizar un algoritmo como el siguiente para hallar la participación de una empresa en otra.

Algoritmo para hallar participaciones

A) *DATOS*

T1 y T2: Tablas relacionales para resultados intermedios. Inicialmente vacías. Tienen las mismas columnas que PARTICIPACIONES.

NORI: Identificador de la empresa cuyo capital deseamos analizar para saber qué parte de él pertenece a otra.

NDES: Identificador de esta última.

RESU: Resultado buscado. Es decir, porcentaje del capital de NORI que pertenece a NDES. Inicialmente a ceros.

INCR: Campo de trabajo. Almacena cuánto se incrementa R en cada nueva iteración.

NITE: Número de iteraciones del algoritmo. Se utiliza para poner un límite al tiempo de ejecución. Inicialmente a ceros.

NFTL: Campo de trabajo. Se utiliza para almacenar el número de filas de una tabla.

B) *PROCESO*

1) Extraer de PARTICIPACIONES todas las filas tales que #EMP.1 = NORI y almacenarlas en T1.
2) Si T1 está vacía, fin del proceso. Si no, seguir.
3) Sumar 1 a NITE. Si NITE = 1000, fin del proceso. Si no, seguir.
4) Borrar el contenido de T2, en caso de que no esté vacía.
5) Almacenar el contenido de T1 en T2.
6) Si no hay en T2 ninguna fila cuya columna #EMP.2 sea igual a NDES, ir al paso (7). Si sí hay, hacer lo siguiente:

— Sumar la columna P de T2 para todas las filas que tengan #EMP.2=NDES y almacenar el resultado en INCR.
— Sumar INCR a RESU.
— Continuar con el paso siguiente.

7) Hallar la suma de los porcentajes, columna P de la tabla T2, para todas sus filas y almacenarla en INCR.
8) Si INCR es menor que 0,1, fin del proceso. Si no, seguir.
9) Borrar el contenido de T1.
10) Obtener la yunción de las tablas T2 y PARTICIPACIONES, con la condición de que #EMP.2 de T2 sea igual a #EMP.1 de PARTICIPACIONES. Por cada fila de la yunción almacenar otra en T1 de manera que: #EMP.1 de T1 reciba el valor de #EMP.1 de T2, #EMP.2 de T1 reciba el valor de #EMP.2 de PARTICIPACIONES. y P de T1 reciba el valor resultante de multiplicar P de T2 por P de PARTICIPACIONES.
11) Volver al paso (2).

Si hay en el grafo bucles con 100 % en todos sus arcos, el proceso tendría un número infinito de iteraciones. Para evitar esto se ha utilizado un contador (NITE) del número de iteraciones con un límite predefinido.

En el Apéndice se incluye un ejemplo de codificación del algoritmo anterior usando los lenguajes PL/I y SQL.

Ejemplo 6. B. de D. para Reservas Ferroviarias

Una Compañía de Ferrocarriles desea implantar un sistema de reservas de plazas para viajeros. El Departamento Comercial de la Compañía describe la estructura de información del transporte de viajeros mediante las definiciones y conceptos siguientes:

Tren

Un tren está formado por varios coches que se desplazan juntos, con origen y destino comunes y con las mismas paradas.

Cada tren tiene un número identificador, único dentro de cada fecha. Es decir, en una fecha dada no puede haber dos trenes con el mismo identificador.

Un identificador puede repetirse en distintas fechas, pero siempre corresponde a trenes con idénticas paradas y horario. Además, se pretende que el identificador del tren, conocido por los clientes a través de las Guías de Ferrocarriles, evoque una imagen de servicio determinada, por lo que para asignar el mismo identificador a dos trenes en distintas fechas, además de tener iguales paradas y horario, deberán tener algunas características de servicio comunes definidas por la política comercial de la Compañía (por ejemplo, camas, restaurante, traslado de automóviles, etc.).

Composición de un tren

Los coches que forman un tren no cambian durante su desplazamiento. Es decir, ni se le añaden ni se le quitan coches entre su origen y su destino. Si esto ocurriera, se considera que se forma un nuevo tren, con un nuevo identificador.

Lo mismo ocurriría si dos trenes que confluyen en una estación juntan sus coches a partir de ella. Se supone que se trata de un tren nuevo, con un identificador diferente. También puede darse el caso inverso: un tren que se divide en dos. Se supone que éstos son trenes distintos, con sus correspondientes identificadores diferentes.

Dentro de un tren, cada coche está identificado por un número único. Además cada coche tiene un número identificador propio, independiente de los trenes en que se enganche, que lo identifica dentro del parque total de coches de la Compañía.

Dos trenes con el mismo identificador, en fechas diferentes, pueden tener distintas composiciones. Por ejemplo, uno puede tener más coches que otro, en función de la demanda de plazas.

Tipos de coches

Cada tipo o modelo de coche tiene un número determinado de plazas de distintas características (camas, literas, primera clase, segunda, etc.). Vamos a simplificar suponiendo que este número es fijo (no siempre es así pues hay coches en los que algunas plazas pueden transformarse de una clase a otra).

Cada plaza de un determinado tipo de coche tiene un número que la identifica dentro del coche.

Tramo

Es el recorrido entre dos paradas sucesivas de un tren.

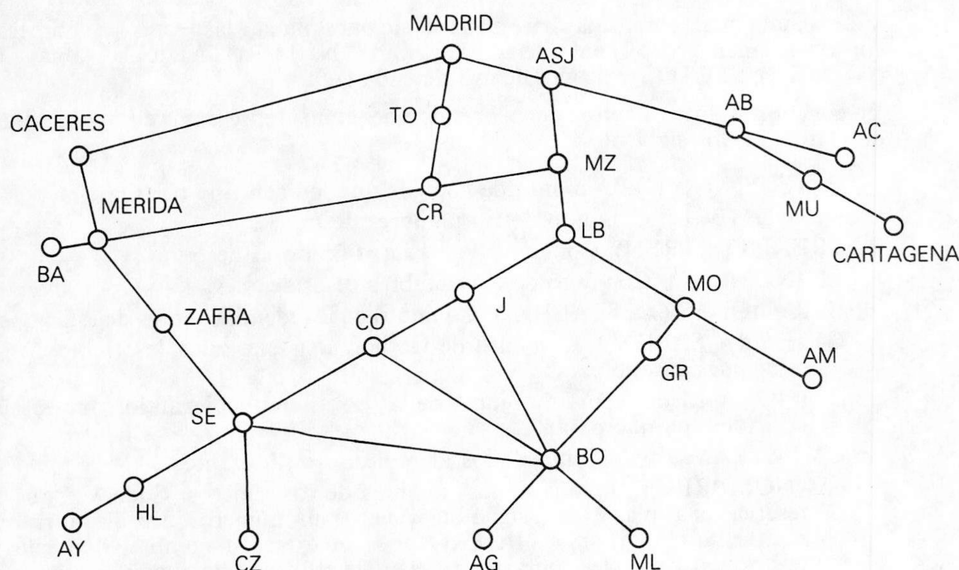
Se identifica con un número único dentro del recorrido correspondiente a un identificador de tren.

Rutas

La Compañía desea establecer un servicio de información a clientes que permita a éstos preguntar por los itinerarios más convenientes para ir de un origen a un destino dados. Esto permite además ofrecer rutas alternativas en caso de que en alguna no haya plazas.

La consulta de Rutas, como la Reserva de Plazas, desea hacerse en tiempo real, lo que puede dar lugar a tiempos de respuesta muy lentos, por el gran número de combinaciones de trenes, estaciones y horarios que pueden presentarse. Por ello se decide hacer un programa que, tratando en diferido, no en tiempo real, todos los trenes y horarios, analice todos los trayectos posibles de todas las estaciones con todas, y elija los mejores. El criterio de optimización puede ser minimizar el número de transbordos o la duración del viaje. Una vez elegidos los trayectos óptimos para todas las parejas de estaciones (origen-destino) posibles, se almacenan en una tabla que llamaremos Tabla de Rutas, que se utiliza para responder a la consulta en tiempo real.

Veamos un ejemplo de consulta de Rutas. Para fijar ideas, supongamos que la Tabla de Rutas contiene información sobre la red española de ferrocarriles, y que



un cliente deseara saber qué rutas hay para ir de Zafra a Cartagena (véase figura superior). Para responder a esta consulta hay que extraer las filas de la Tabla de Rutas que tienen origen y destino en estas estaciones. Podrían ser las siguientes:

Origen ruta	*Destino* ruta	*Número* ruta	*Trayecto* De	A	*Número* tren	*Salida* Hora	Día	*Llegada* Hora	Día
Zafra	Cartagena	1	Zafra	Mérida	A2320	20.25	D	21.18	D
Zafra	Cartagena	1	Mérida	Madrid	TR593	07.45	D+1	13.47	D+1
Zafra	Cartagena	1	Madrid	Cartagena	T124	15.45	D+1	21.28	D+1
Zafra	Cartagena	2	Zafra	Cáceres	A2320	20.25	D	22.30	D
Zafra	Cartagena	2	Cáceres	Madrid	F331	03.32	D+1	08.25	D+1
Zafra	Cartagena	2	Madrid	Cartagena	R520	08.45	D+1	15.33	D+1

En este ejemplo se ofrecen dos rutas. Con la Ruta 2 se llega antes que con la Ruta 1, pero hay que hacer noche en el tren y esperar varias horas de madrugada en Cáceres. El cliente podrá elegir la que le convenga más.

Esta tabla puede ser demasiado grande para tenerla en memoria. Para una red ferroviaria de 2.000 estaciones, al combinar todas con todas resultan 4.000.000 de posibilidades. Suponiendo una media de tres rutas por cada una y tres trayectos por ruta, tendríamos unos 36.000.000 de entradas en la tabla. Si cada entrada necesitara 50 octetos, la tabla ocuparía unos 1,8 Gigaoctetos. Esto nos lleva a almacenar la tabla en disco, como una tabla relacional cuyo esquema es:

RUTAS (–ORIGEN.RUTA, DESTINO.RUTA, NUM.RUTA, NUM.TRAYECTO–, ORIGEN.TRAYECTO, NUM.TREN, HORA.SALIDA, FECHA.SALIDA, HORA.LLEGADA, FECHA.LLEGADA).

Suponemos que, para una pareja de estaciones dada, las rutas se numeran consecutivamente de 1 en adelante (atributo NUM.RUTA de la tabla). El (NUM.TRAYECTO) se asigna dentro de cada ruta.

En conclusión, los conjuntos de valores que deseamos representar en nuestra Base de Datos son los siguientes:

C1 (TIPO_TREN): Conjunto de códigos que indican tipo de tren.

C2 (NOMBRE): Conjunto de los nombres de trenes.

TRENES (#TREN): Conjunto de los identificadores de trenes.

C3 (NOMBRE): Conjunto de los nombres de estaciones.

ESTACIONES (#ESTACION): Conjunto de los identificadores de estaciones.

C4 (HORA_SALIDA): Conjunto de las horas en que salen los trenes de las estaciones en que paran.

C5 (HORA_LLEGADA): Conjunto de las horas en que llegan los trenes a las estaciones en que paran.

CA (#TRAMO): Conjunto de los identificadores de tramos.

C7 (NOMBRE): Conjunto de los nombres de los viajeros. Cada vez que se reserva una plaza, se asigna al viajero un número identificador, que representaremos por #VIAJERO, y se guardan su nombre, domicilio y teléfono para poder avisarle en caso de cambios de fuerza mayor que afecten a su reserva.

COCHES (#COCHE): Conjunto de los identificadores de coches.

C8 (TIPO_COCHE): Conjunto de códigos que indican el tipo de coche.

C9 (#COCHE_EN_TREN): Conjunto de números que significan número de coche dentro de los trenes.

C10 (FECHA): Conjunto de fechas.

C11 (TELEFONO): Conjunto de los números de teléfono de los viajeros que tienen plaza reservada.

Si un viajero no tiene teléfono, o no lo conocemos, le asignaremos el número 0.

C12 (DOMICILIO): Domicilios de los viajeros que tienen plaza reservada.

C13 (#PLAZA): Conjunto de los identificadores de plazas dentro de los coches.

C14 (TIPO_PLAZA): Conjunto de códigos que indican las características de las plazas de los coches (cama, litera, primera o segunda clase, fumador o no, etc.).

C15 (#VIAJERO): Conjunto de los identificadores asignados a los viajeros.

TRAMOS (#TREN, #TRAMO): Conjunto de parejas de valores que asocian cada tren con sus tramos.

RUTAS (–#ESTACION.1, #ESTACION.2, #RUTA, #TRAYECTO–, #ESTACION.3, #TREN, HORA_SALIDA, FECHA_SALIDA, HORA_LLEGADA): Tabla de Rutas para consultas en tiempo real.

Significado de los atributos:

(#ESTACION.1): Estación origen de la ruta.

(#ESTACION.2): Estación de destino de la ruta.

(#RUTA): Número identificador de la Ruta dentro de cada pareja de estaciones (origen-destino).

(#TRAYECTO): Número de trayecto dentro de la ruta.

(#ESTACION.3): Estación origen del trayecto.

(#TREN): Identificador del tren.

PLAZAS_COCHE (TIPO_COCHE, #PLAZA): Contiene las plazas de cada tipo de coche.

COMPOSICION (#TREN, FECHA, #COCHE): Contiene la composición de cada tren.

Significa que el coche #COCHE forma parte del tren #TREN que sale en el día FECHA.

ENGANCHES (#TREN.1, FECHA.1, #COCHE, #TREN.2, FECHA.2): Nos indica qué coches pasan de un tren a otro, con los viajeros a bordo.

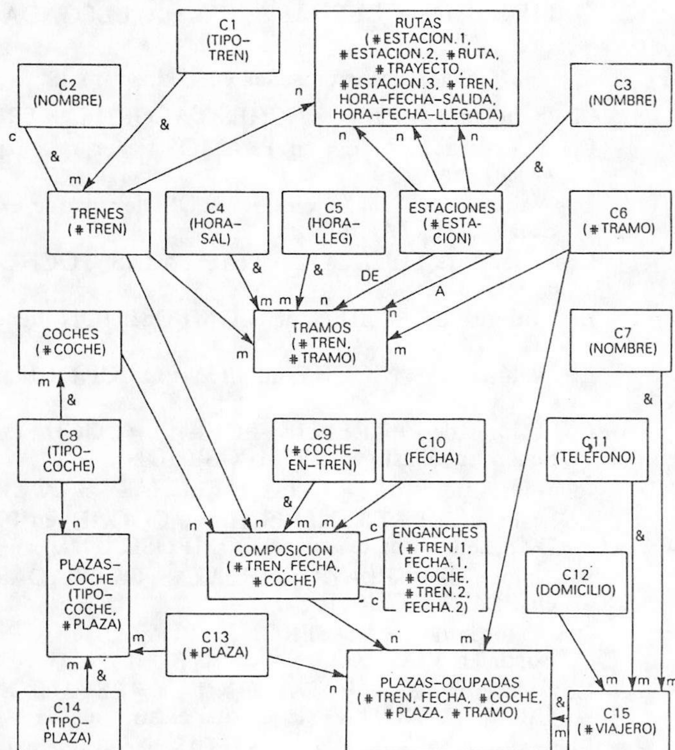
Significa que si un viajero tiene plaza reservada en el coche #COCHE para el tren #TREN.1 que sale en el día FECHA.1, no necesita transbordar para cambiar al tren #TREN.2 que sale en el día FECHA.2.

PLAZAS_OCUPADAS (#TREN, FECHA, #COCHE, #PLAZA, #TRAMO): Nos indica qué plazas están reservadas y en qué tramos.

Entre ESTACIONES y TRAMOS hay una asociación explícita que indica qué estación es origen de cada tramo. La llamamos «DE».

Hay otra asociación explícita que indica qué estación es destino en cada tramo. La llamamos «A».

Los conjuntos anteriores se representan en un diagrama RE/R inicial como el mostrado en la figura siguiente.



El esquema de diseño obtenido es el siguiente:

Esquema de diseño

1) Tablas:

TRENES (–#TREN–, –NOMBRE–, TIPO_TREN). Clave primaria: #TREN.

ESTACIONES (–#ESTACION–, NOMBRE–). Clave primaria: #ESTACION.

TRAMOS (–#TREN, #TRAMO–, DE, A, HORA_SAL, HORA_LLEG).

COCHES (–#COCHE–, TIPO_COCHE).

PLAZAS_COCHE (–TIPO_COCHE, #PLAZA–, TIPO_PLAZA).

COMPOSICION (–#TREN, FECHA, #COCHE–, #COCHE_EN_TREN).

ENGANCHES (–#TREN.1, FECHA.1, #COCHE–, #TREN.2, FECHA.2).

PLAZAS_OCUPADAS (–#TREN, FECHA, #COCHE, #PLAZA, #TRAMO–, #VIAJERO).

VIAJEROS (–#VIAJERO–, NOMBRE, TELEFONO, DOMICILIO).

RUTAS (–#ESTACION.1, #ESTACION.2, #RUTA, #TRAYECTO–, #ESTACION.3, #TREN, HORA_SALIDA, FECHA_SALIDA, HORA_LLEGADA, FECHA_LLEGADA).

2) Condiciones adicionales expresadas en el diagrama:

Clave alternativa en ENGANCHES: (#COCHE, #TREN.2, FECHA.2).

Los valores de #TREN en TRAMOS tienen que existir en TRENES, y viceversa.

Los valores de DE y A en TRAMOS tienen que existir en #ESTACION de ESTACIONES.

Los valores de TIPO_COCHE en PLAZAS_COCHE tienen que existir en COCHES.

Los valores de #TREN en COMPOSICION tienen que existir en TRENES.

Los valores de #COCHE en COMPOSICION tienen que existir en COCHES

Los valores de (#TREN.1, FECHA.1, #COCHE) de ENGANCHES tienen que existir en COMPOSICION.

Idem los valores de (#TREN.2, FECHA.2, #COCHE).

Los valores de (#TREN, FECHA, #COCHE) en PLAZAS_OCUPADAS tienen que existir en COMPOSICION.

Los valores de (#PLAZA) en PLAZAS_OCUPADAS tienen que existir en PLAZAS_COCHE.

Los valores de (#VIAJERO) en PLAZAS_OCUPADAS tienen que existir en VIAJEROS, y viceversa.

Los viajeros de (#ESTACION.1), (#ESTACION.2) y (#ESTACION.3) en RUTAS tienen que existir en ESTACIONES.

Los valores de (#TREN) en RUTAS tienen que existir en TRENES.

3) Condiciones adicionales no expresadas en el diagrama:

Claves alternativas en TRAMOS: (#TREN, DE), (#TREN, HORA_SAL), (#TREN, HORA_LLEG).

Clave alternativa en COMPOSICION: (#TREN, FECHA, #COCHE_EN_TREN).

Claves alternativas en RUTAS: (#ESTACION.1, #ESTACION.2, #RUTA, #ESTACION.3), (#ESTACION.1, #ESTACION.2, #RUTA, #TREN).

Los valores de (#TREN, #TRAMO) en PLAZAS_OCUPADAS tienen que existir en TRAMOS.

Ejemplo 7. B. de Datos de Clientes de un Banco

Los conjuntos de valores a almacenar en la Base de Datos son los siguientes:

C1 (NOMBRE): Conjunto de nombres de los clientes.

C2 (DOMICILIO): Conjunto de domicilios de los clientes.

C3 (FECHA_NAC): Conjunto de las fechas de nacimiento de los clientes.

C4 (TIPO_CLIENTE): Códigos de tipos de clientes.

C5 (PROFESION): Códigos de profesiones.

C6 (SEXO): Códigos de sexo.

CLIENTES (#CL): Conjunto de los identificadores de los clientes.

REL_CLIENTES (#CL.1, #CL.2): Conjunto de parejas de identificadores de clientes que están relacionados por algún criterio (por ejemplo, por ser familiares, por ser socios, etc.).

C7 (TIPO_REL): Códigos de los tipos de relación que puede haber entre dos clientes.

GRUPOS (#GRUPO): Conjunto de los identificadores de grupos de clientes.

Algunos clientes forman unidades con otros, a efectos de ciertas operaciones con el Banco. Estas unidades las llamamos grupos. Un grupo puede estar formado por varios clientes. Un cliente sólo puede pertenecer a un grupo. Ejemplo: un grupo de empresas (matriz y filiales) cuyas operaciones de riesgo con el Banco se negocian globalmente.

Si un cliente no pertenece a ningún grupo, le asignaremos el grupo 0.

C11 (DESGRU): Conjunto de descripciones o nombres de los grupos.

PRODUCTOS (#PR): Conjunto de identificadores de todos los productos y servicios ofrecidos por el Banco.

C10 (TIPO_PROD): Códigos de tipos de productos (por ejemplo, cuenta corriente, libreta de ahorros, imposición a plazo fijo, tarjeta de débito, etc.

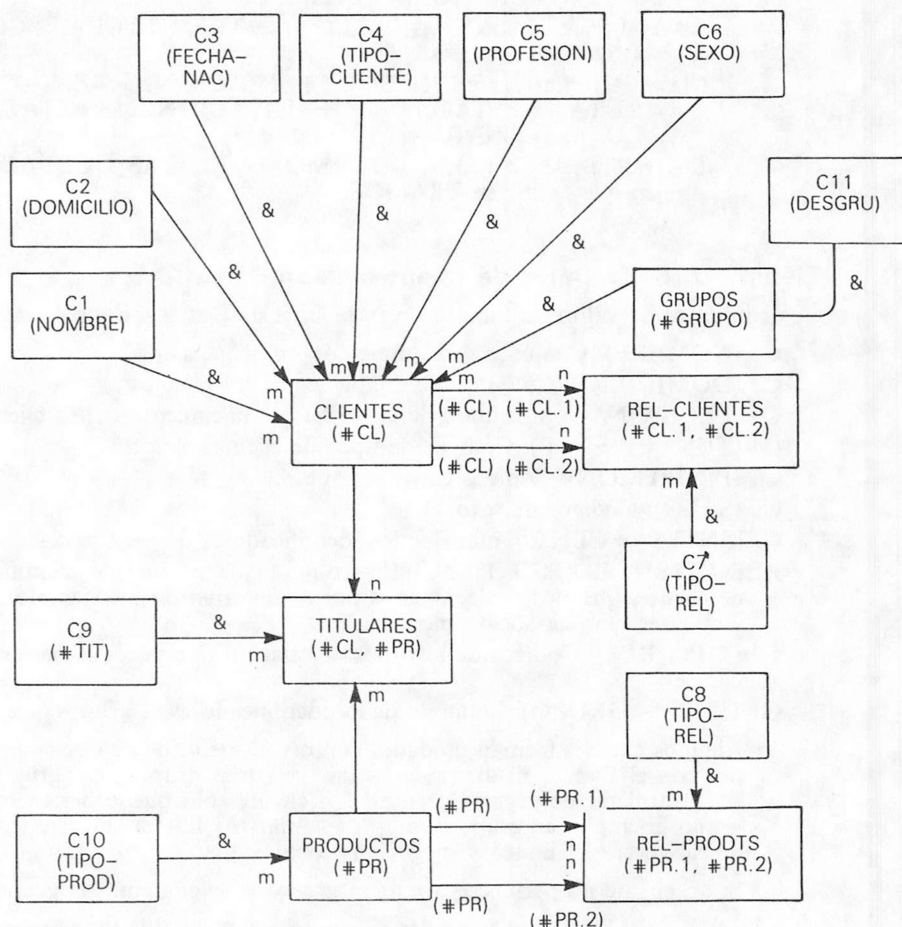
REL_PRODTS (#PR.1, #PR.2): Conjunto de parejas de identificadores de productos que están relacionados por algún criterio (por ejemplo, una cuenta de crédito con la cuenta corriente de donde se cobran los intereses).

C8 (TIPO_REL): Códigos de los tipos de relación que puede haber entre los productos.

TITULARES (#CL, #PR): Asociación que indica qué clientes son titulares de qué productos.

C9 (#TIT): Conjunto de los números de orden de los titulares dentro de cada producto.

Los conjuntos anteriores se representan en un diagrama RE/R inicial como el mostrado en la figura siguiente.



El esquema de diseño obtenido es el siguiente:

Esquema de diseño

1) Tablas:

CLIENTES (–#CL–, NOMBRE, DOMICILIO, FECHA_NAC, TIPO_CLIENTE, PROFESION, SEXO, #GRUPO).
GRUPOS (–#GRUPO–, –DESGRU–). Clave primaria: #GRUPO.
REL_CLIENTES (–#CL.1, #CL.2–, TIPO_REL).
PRODUCTOS (–#PR–, TIPO_PROD).
REL_PRODTS (–#PR.1, #PR.2–, TIPO_REL).
TITULARES (–#CL, #PR–, #TIT).

2) Condiciones adicionales expresadas en el diagrama:

Los valores de #GRUPO en CLIENTES tienen que existir en GRUPOS, y viceversa.

Los valores de (#CL.1) y (#CL.2) en REL_CLIENTES tienen que existir en CLIENTES.

Los valores de (#PR.1) y (#PR.2) en REL_PRODTS tienen que existir en PRODUCTOS.

Los valores de #CL en TITULARES tienen que existir en CLIENTES.

Los valores de #PR en TITULARES tienen que existir en PRODUCTOS, y viceversa.

3) Condiciones adicionales no expresadas en el diagrama:

Clave alternativa en TITULARES: (#PR, #TIT).

Ejemplo 8. B. de D. Universitaria

Un centro que ofrece clases de nivel universitario funciona con las siguientes normas.

- Los alumnos se matriculan en las asignaturas que desean, siempre que respeten los prerrequisitos entre éstas. Así, por ejemplo, antes de poder cursar la asignatura Matemáticas-2 hay que haber aprobado Matemáticas-1.
- Las clases admiten un máximo de 35 alumnos. Por ello, los alumnos matriculados en una asignatura se dividen en grupos, con un máximo de 35 alumnos por grupo.
- Los alumnos de un grupo asisten a las mismas clases, con el mismo horario.
- Todos los grupos de una asignatura siguen el mismo programa, con los mismos textos y exámenes.
- Cada grupo sólo recibe una clase al día por asignatura.
- Las clases de un grupo de una asignatura las da un solo profesor. Otros grupos de la misma asignatura pueden tener profesores diferentes.
- A cada alumno se le asigna un profesor como tutor personal.
- Los alumnos son calificados en todos los exámenes de las asignaturas en que están matriculados (la calificación es de 0 a 10, o «no presentado»).

Deseamos diseñar una Base de Datos que contenga los conjuntos de valores siguientes:

C1 (NOMBRE): Conjunto de nombres de las asignaturas.

C5 (LIBRO): Conjunto de títulos de todos los libros de texto que se usan en el Centro.

C4 (#ASIG): Conjunto de identificadores de asignaturas.

PREREQ (#ASIG.1, #ASIG.2): Asociación entre asignaturas. Significa que la asignatura #ASIG.1 es prerrequisito para matricularse en la #ASIG.2.

C3 (#GRUP): Conjunto de números de grupo. A cada grupo se le asigna un número que lo identifica dentro de cada asignatura.

GRUPOS (#ASIG, #GRUP): Asociación entre cada asignatura y sus grupos correspondientes.

C6 (DIA): Conjunto de los nombres de los días de la semana.

C7 (HORA): Conjunto de las horas de comienzo de las clases.

C8 (AULA): Conjunto de los identificadores de aulas.

CLASES (#ASIG, #GRUP, DIA): Días de clase por semana de cada grupo.

C9 (#ALUM): Conjunto de los identificadores de alumnos.

C10 (#PROF): Conjunto de identificadores de los profesores.

C11 (NOMBRE): Conjunto de nombres de los profesores.

C12 (SUELDO): Conjunto de los sueldos de los profesores.

C13 (CATEG): Conjunto de códigos que representan categorías o niveles de profesores.

C15 (NOMBRE): Conjunto de nombres de los alumnos.

C16 (AÑO): Conjunto de números que representan los años que cada alumno lleva estudiando en el Centro.

C2 (#EXAM): Cada asignatura tiene sus propios planes de exámenes. Los exámenes se identifican con un número dentro de cada asignatura. C2 es el conjunto de estos números.

CALIFICACS (#ASIG, #EXAM, #ALUM): Asocia cada alumno con los exámenes que le corresponden.

C14 (NOTA): Conjunto de las notas obtenidas por los alumnos en los exámenes.

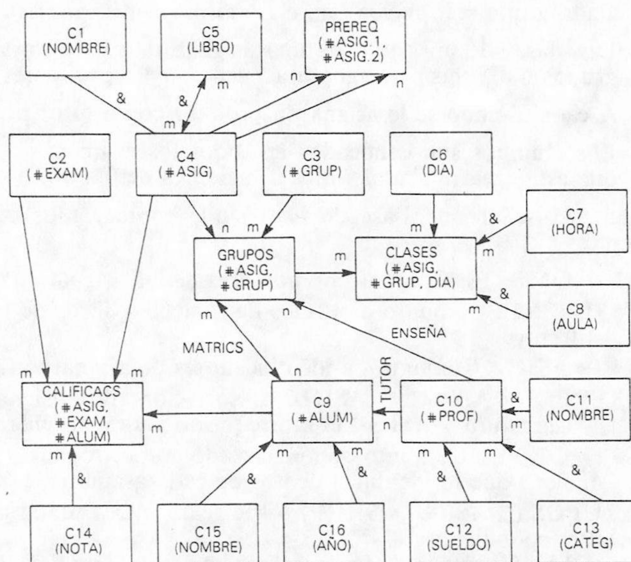
Además tenemos las siguientes asociaciones explícitas:

ENSEÑA: Asociación entre C10 y GRUPOS. Asocia cada profesor con las asignaturas y grupos donde dé clase.

TUTOR: Asociación entre C10 y C9. Asocia cada profesor con los alumnos de que es tutor.

MATRICS: Asociación entre C9 y GRUPOS. Asocia cada alumno con los grupos en que está mactriculado.

El diagrama RE/R de partida es el siguiente:



Se obtiene el siguiente esquema de diseño.

Esquema de diseño

1) Tablas:

ASIGNATURAS (-#ASIG-, -NOMBRE-). Clave primaria: #ASIG.
TEXTOS (#ASIG, LIBRO).
PRERREQ (#ASIG. 1, #ASIG. 2).
GRUPOS (-#ASIG, #GRUP-, #PROF).
El #PROF representa al profesor que da las clases en cada grupo.
CLASES (-#ASIG, #GRUP, DIA-, HORA, AULA).
ALUMNOS (-#ALUM-, NOMBRE, AÑO, #PROF).
El #PROF representa al profesor que es tutor de cada alumno.
MATRICS (#ASIG, #GRUP, #ALUM).
PROFESORES (-#PROF-, NOMBRE, CATEG, SUELDO).
CALIFICACS (-#ASIG, #EXAM, #ALUM-, NOTA).

2) Condiciones expresadas en el diagrama:

Los valores de #ASIG en TEXTOS tienen que existir en ASIGNATURAS, y viceversa.

Los valores de (#ASIG. 1) y (#ASIG.2) en PRERREQ tienen que existir en ASIGNATURAS.

Los valores de (#ASIG) en GRUPOS tienen que existir en ASIGNATURAS.

Los valores de (#PROF) en GRUPOS tienen que existir en PROFESORES.

Los valores de (#ASIG, #GRUP) en CLASES tienen que existir en GRUPOS, y viceversa.

Los valores de (#PROF) en ALUMNOS tienen que existir en PROFESORES.

Los valores de (#ASIG, #GRUP) en MATRICS tienen que existir en GRUPOS.

Los valores de (#ALUM) en MATRICS tienen que existir en ALUMNOS, y viceversa.

Los valores de (#ASIG) en CALIFICACS tienen que existir en ASIGNATURAS, y viceversa.

Los valores de (#ALUM) en CALIFICACS tienen que existir en ALUMNOS, y viceversa.

3) Condiciones no expresadas en el diagrama:

Los valores de (#ASIG, #ALUM) en CALIFICACS tienen que existir en MATRICS, y viceversa.

Los prerrequisitos entre asignaturas no pueden tener «bucles». Es decir, una asignatura no puede ser prerrequisito de sí misma, directa o indirectamente. Esto se aplica a la tabla PRERREQ.

6

EJERCICIOS PROPUESTOS

En los ejercicios siguientes, para simplificación de los dibujos, dejaremos a veces las asociaciones explícitas sin nombre. En este caso indicaremos que son explícitas adjuntándoles el signo &. También, por la misma razón, se omitirán a veces los nombres de los conjuntos de valores representados en los rectángulos.

1) *Sean los conjuntos de valores siguientes:*

C1 (NE): Nombres de los empleados de una empresa.

C2 (#D): Números identificadores de los departamentos de la empresa.

C3 (ND): Nombres de los departamentos de la empresa.

C4 (#T): Números de teléfono, dentro de la empresa, asignados a empleados.

C5 (CJ): Código de si un empleado es jefe o no. Es un conjunto de dos valores: S (si es jefe), N (si no es jefe).

Todos los empleados tienen nombres diferentes.

Un departamento puede tener varios empleados o ninguno. Un empleado tiene que estar en un departamento y no puede pertenecer a más de uno.

Un departamento no puede tener más de un jefe, aunque puede estar sin jefe en algún momento.

Cada empleado tiene un teléfono. Si es jefe, lo usa en exclusiva, si no, comparte su uso con otros empleados no jefes.

Representar la descripción anterior en un diagrama y obtener el esquema de diseño.

2) *Sean los conjuntos de valores siguientes:*

C1 (#M): Matrículas de los coches de una empresa asignados a sus vendedores.

C2 (MO): Modelos de estos coches.

C3 (#V): Identificadores de vendedores.

C4 (NV): Nombres de vendedores.

La empresa dispone de una flota de coches para sus vendedores. A cada vendedor se le asigna un coche, y cada coche sólo se asigna a un vendedor. Sea R (#M, #V) la asociación entre vendedores y coches.

Representar esta descripción en un diagrama y obtener el esquema de diseño.

3) La Regla de Propagación de Claves, tal como se enunció en el capítulo correspondiente, se aplica entre dos rectángulos ligados por una asociación explícita de cardinalidad (1 : n). *Demostrar que también puede aplicarse cuando la cardinalidad es (c : n) si permitimos que el atributo propagado tome valores nulos.*

4) Queremos diseñar una Base de Datos para almacenar información sobre los Departamentos, Empleados y Oficinas de una empresa. Sus identificadores respectivos son #D, #E y #O. Por cada Departamento se desea almacenar su presupuesto (lo designaremos por PD) en pesetas, y el número identificador del empleado que lo dirige. También se desea guardar por Departamento los identificadores de sus empleados, los de las oficinas que ocupa y los Proyectos de los que cada Departamento es responsable. Los proyectos tienen un identificador llamado #P. Para cada empleado se desea saber los proyectos a los que está asignado, la oficina donde está su puesto de trabajo, y por tanto donde recibe su correspondencia y su número de teléfono en dicha oficina. Cada proyecto tiene un presupuesto, llamado PP, en pesetas, no incluido en el del Departa-

mento responsable del proyecto. Cada oficina tiene una superficie en metros cuadrados que se desea almacenar también, con el nombre SO. Para cada empleado se desea guardar una información histórica de los aumentos salariales que ha tenido, independientemente de si han ido acompañados de ascenso o no. Por cada subida de sueldo se guardará la fecha en que se ha producido (la llamaremos FS), la categoría profesional del empleado en ese momento (CP) y el nuevo sueldo en pesetas (NS). También se desea saber para cada oficina sus números de teléfono (los números de teléfono los designaremos como #T). Un Departamento puede tener varios empleados, proyectos y oficinas, pero a cada uno de éstos sólo corresponde un Departamento. Un teléfono está en una oficina. Una oficina puede tener varios teléfonos.

Obtener el esquema relacional de diseño de la Base de Datos de acuerdo con esta descripción, completándola si hace falta con las hipótesis adicionales que se crean razonables.

5) Queremos diseñar una Base de Datos para almacenar información sobre nuestros Clientes y sus Pedidos, cuyos identificadores respectivos son #C y #P. Los artículos comercializados por nuestra organización tienen un identificador llamado #A. Algunos son fabricados en factorías propias, cuyo identificador es #F, y otros hay que adquirirlos en proveedores externos. Cada cliente dispone de un crédito cuyo límite se llama LC, de un descuento llamado DC por compras masivas, y de un saldo, SC, cuyo importe varía conforme va pagando los pedidos entregados. Los pedidos se entregan en una dirección de entre varias posibles de cada cliente. Estas direcciones se llaman DE (dirección de entrega). Los pedidos se componen de dos partes: una cabecera de pedido y varias líneas de detalle. En la cabecera figuran la fecha del pedido y el cliente que lo hace. Las líneas de detalle contienen el artículo pedido y la cantidad pedida. Se desea guardar toda esta información en la Base de Datos. Cada artículo tiene una descripción única. Se desea también llevar un control de las necesidades de fabricación. Para ello, por cada fábrica propia se guardará el nivel de inventario en el almacén de la fábrica de cada artículo (CD: Cantidad Disponible), y el nivel mínimo de existencias a partir del cual hay riesgo de quedarse sin existencias (NR: Nivel de Riesgo). Un pedido puede servirse en varias entregas. Para llevar un control de éstas, se guardará por cada línea de detalle un contador con lo que queda pendiente de entregar en cada momento, que se llama PE.

Obtener el esquema relacional de diseño de la Base de Datos de acuerdo con esta descripción, completándola si hace falta con las hipótesis adicionales que se crean razonables.

APENDICE



Bibliografía

Existe una abundante bibliografía sobre los temas tratados en este libro. Se incluye a continuación una selección de títulos en los que el lector interesado podrá ampliar sus conocimientos. Además, para los interesados en seguir la evolución del estado del arte, se recomienda recurrir a las publicaciones periódicas especializadas, entre las que cabe destacar Transactions on Database Systems de ACM.

Date, C. J. «An Introduction to Database Systems». Vol. I. Third edition. Reading, Massachusetts: Addison-Wesley (1981).

Date, C. J. «An Introduction to Database Systems». Vol. II. Reading, Massachusetts: Addison-Wesley (1983).

Date, C. J. «Relational Database: Selected Writings». Reading, Massachusetts: Addison-Wesley (1986).

Deaño, Alfredo. «Introducción a la lógica formal». Madrid: Alianza Editorial (1980).

Delobel, Claude; Adiba, Michel. «Bases de donées et Systèmes Relationnels». París: Dunod (1982).

Fernández Baizán, Covadonga. «El modelo relacional de datos: de los fundamentos a los modelos deductivos». Madrid: Díaz de Santos (1987).

Fernández Fernández, Gregorio. «Fundamentos de los ordenadores. Vol. II: Lenguajes». Madrid: Departamento de Publicaciones de la Escuela Técnica Superior de Ingenieros de Telecomunicación (1978).

Flavin, Matt. «Fundamental Concepts of Information modeling». New York: Yourdon (1981).

Gardarin, Georges. «Bases de datos». Madrid: Paraninfo (1987).

Gardarin, G.; Valduriez, P. «Bases de Données Relationnelles: Analyse et comparaison des systèmes». París: Eyrolles (1985).

Hubbard, George U. «Computer Assisted Data Base Design». New York: Van Nostrand Reinhold (1981).

Kent, William. «Data and Reality». Amsterdam: North Holland (1984).

Maier, David. «The theory of Relational Databases». London: Pitman (1983).

A

Miranda, S. M.; Busta, J. M. «L'art des Bases de Données, 2: Les Bases de Données Relationnelles». París: Eyrolles (1986).

Mounin, Georges. «Claves para la lingüística». Barcelona: Ed. Anagrama (1968).

Rivero Cornelio, Enrique. «Fundamentos de los ordenadores. Vol. I: Representación y codificación de la información». Madrid: Departamento de Publicaciones de la Escuela Técnica Superior de Ingenieros de Telecomunicación (1978).

Sundgren, Bo. «Theory of Data Bases». New York: Petrocelli Charter (1975).

Tsichritzis, Dionysios C.; Lochovsky, Frederick, H. «Data Models». Englewood Cliffs, New Jersey: Prentice-Hall (1982).

Ullman, Jeffrey D. «Principles of Database Systems». Second edition. London: Pitman (1982).

Vetter, M.; Maddison, R. N. «Database Design Methodology». London: Prentice-Hall International (1981).

Yao, S. Bing. «Principles of Data Base Design». Vol. I: «Logical organizations». Englewood Cliffs, N. J.: Prentice Hall (1985).

APENDICE

Recordatorio de la notación utilizada



D: Conjunto de dominios.

⊆: Operador entre conjuntos que significa «contenido en» o «igual a».

∅: Conjunto vacío.

R: Conjunto de todas las relaciones posibles construidas sobre **D**.

∪: Unión de conjuntos.

−: Diferencia de conjuntos.

∩: Intersección de conjuntos.

P(A) (R): Proyección de la relación R sobre su atributo A (leer: Proyección sobre A de R).

R[A]: Proyección de la relación R sobre su atributo A.

S(F) (R): Selección según la fórmula F sobre la relación R.

R(X=x, Y): Conjunto de todos los valores de Y que están en las tuplas que tengan X=x en la relación R.

R/S: Cociente entre las relaciones R y S.

⊂: Operación entre conjuntos. Significa «contenido en».

C: Operador de comparación, en general.

Y: Yunción de relaciones.

R **Y** S(i **C** j): Yunción de las relaciones R y S con la condición **C** entre los atributos i de R y j de S.

R **Y** S: Yunción natural entre las relaciones R y S.

R * S: Yunción natural entre las relaciones R y S.

YP: Yunción posibilista («may-be-join»).

YE: Yunción externa.

UE: Unión externa.

TUP: Conjunto de todas las tuplas posibles formadas con los dominios de **D**.

DOM(t): Conjunto en el que toma valores una variable t.

t(1): Primer componente de la tupla t.

∃: Cuantificador de existencia («existe un»).

∀: Cuantificador universal («para todo»).

CINT: Conjunto de condiciones de integridad.

DF: Conjunto de Dependencias Funcionales.

DF+: Clausura de **DF**.

→: Dependencia Funcional.

→→: Dependencia Plural (multivaluada).

DEP: Conjunto de Dependencias Funcionales y Plurales.

DEP+: Clausura de **DEP**.

DY: Dependencia Yuncional («join dependency»).

DG: Dependencia Generalizada.

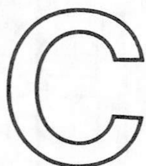
DG: Conjunto de Dependencias Generalizadas.

∧: Operación lógica «Y» (AND).

∨: Operación lógica «O» (OR).

¬ Operación lógica de negación (NOT).

APENDICE



Operadores lógicos

Los representamos por ∧ (operador «Y», o de conjunción), ∨ (operador «O», o de disyunción) y ¬ (operador «NO», o de negación).

Estos operadores se aplican a operandos que son variables o expresiones que pueden tomar uno de dos valores posibles: o el valor «V» (Verdadero) o el valor «F» (Falso). El resultado de aplicar los operadores también puede tomar uno de estos dos valores.

Los operadores ∧ y ∨ se aplican a dos operandos. El operador ¬ se aplica a un solo operando.

El resultado de ∧ es «V» si los dos operandos a los que se aplica tienen el valor «V». En caso contrario, el resultado es «F». Esto se expresa en la siguiente tabla:

∧	V	F
V	V	F
F	F	F

El resultado de ∨ es «V» si uno cualquiera de sus dos operandos, o ambos, tienen el valor «V». Esto se expresa en la siguiente tabla:

∨	V	F
V	V	V
F	V	F

El resultado de ¬ es «V» si su operando tiene el valor «F». En caso contrario, el resultado es «F».

Ejemplos:

Consideremos el conjunto de los empleados de una empresa. Sea E una variable que toma valores en el conjunto de edades de los empleados, y S otra variable

que toma valores en el conjunto de los sueldos de los empleados. Podemos construir las expresiones siguientes:

- (E > 30).

  Esta expresión tomará el valor «V» para todos los empleados de más de 30 años y «F» para el resto.

- (S > 200).

  Esta expresión tomará el valor «V» para todos los empleados que ganen más de 200.000 pesetas.

- $(E > 30) \wedge (S > 200)$.

  Tomará el valor «V» para todos los empleados que cumplan a la vez ambas condiciones. Es decir, para los que tengan más de 30 años y ganen más de 200.000 pesetas.

- $(E > 30) \vee (S > 200)$.

  Será «V» para todos los empleados que cumplan una cualquiera de las dos condiciones. Es decir, para todos los que, o bien tengan más de 30 años, o bien ganen más de 200.000 pesetas, o bien ambas cosas a la vez.

- $\neg$ (E > 30).

  Será «V» para todos los empleados en los que la condición (E > 30) sea «F». Es decir, para todos los que no tengan más de 30 años.

- $((E > 30) \wedge (S > 200)) \vee (S > 300)$.

  Será «V» para los empleados que o bien ganen más de 300.000 pesetas, o bien ganen más de 200.000 y tengan más de 30 años.

APENDICE

D Programa para relaciones entre empresas

```
PARTEM01 : PROC OPTIONS (MAIN) ;

/* ESTE PROGRAMA CALCULA EL PORCENTAJE DEL CAPITAL DE   */
/*      UNA EMPRESA QUE PERTENECE A OTRA. EL IDENTI-    */
/*      FICADOR DE LA PRIMERA SE LEE EN NORI, Y EL DE   */
/*      LA SEGUNDA EN NDES.                             */

DCL NORI CHAR (4),
    NDES CHAR (4),
    RESU DECIMAL FIXED (6,2),
    INCR DECIMAL FIXED (6,2),
    NITE BIN FIXED,
    NFIL BIN FIXED ;

EXEC SQL INCLUDE SQLCA ;
EXEC SQL INCLUDE PARTICIPACIONES ;
EXEC SQL WHENEVER SQLERROR GO TO MENERR ;

EXEC SQL CREATE TABLE T1
    (NEM1 CHAR (4)          NOT NULL WITH DEFAULT
    ,NEM2 CHAR (4)          NOT NULL WITH DEFAULT
    ,P    DECIMAL (5, 2) NOT NULL WITH DEFAULT
    ) ;

EXEC SQL CREATE TABLE T2
    (NEM1 CHAR (4)          NOT NULL WITH DEFAULT
    ,NEM2 CHAR (4)          NOT NULL WITH DEFAULT
    ,P    DECIMAL (5, 2) NOT NULL WITH DEFAULT
    ) ;


NITE = 0 ;
RESU = 0 ;
INCR = 100 ;
GET DATA (NORI, NDES) ;
PUT DATA (NORI, NDES) ;


EXEC SQL INSERT INTO T1
         SELECT *
         FROM PARTICIPACIONES
         WHERE NEM1 = : NORI ;
```

D

```
DO WHILE  ( INCR > 0.05 ) ;

    EXEC SQL SELECT COUNT(*)
             INTO : NFIL
             FROM T1 ;

    IF NFIL = 0 THEN GO TO FIN;
    NITE = NITE + 1;
    IF NITE = 1000 THEN GO TO FIN;

    EXEC SQL DELETE FROM T2;

 Page       1

    EXEC SQL INSERT INTO T2
             SELECT *
             FROM T1;

    EXEC SQL SELECT COUNT(*)
             INTO : NFIL
             FROM T2
             WHERE NEM2 = : NDES;
   IF NFIL > 0 THEN  DO;
             EXEC SQL SELECT SUM(P)
                      INTO : INCR
                      FROM T2
                      WHERE NEM2 = : NDES;
             RESU = RESU + INCR;
             END;

    EXEC SQL SELECT SUM (P)
             INTO : INCR
             FROM T2 ;

    EXEC SQL DELETE FROM T1;
    EXEC SQL INSERT INTO T1
             SELECT X.NEM1, Y.NEM2, ((X.P) * (Y.P)) / 100
             FROM T2 X, PARTICIPACIONES Y
             WHERE X.NEM2 = Y.NEM1 ;

    END ;


FIN :
    PUT LIST ('NUMERO ITERACIONES :') ;
    PUT DATA (NITE) ;
    PUT LIST ('RESULTADO :') ;
    PUT DATA (RESU) ;
    EXEC SQL DROP TABLE T1;
    EXEC SQL DROP TABLE T2;
    GO TO SALIR ;

MENERR :
    PUT LIST ('ERROR SENTENCIA SQL ') ;
    PUT DATA (SQLCODE) ;
    GO TO SALIR ;

SALIR  :
   END;
Page       2
```

APENDICE

E Programa para diseño

DESCRIPCION GENERAL

Ya hemos comentado previamente la conveniencia de automatizar el proceso de diseño. A modo de ejemplo incluimos aquí una breve descripción funcional de un programa que puede servir de ayuda en la aplicación de las reglas de transformación a un diagrama RE/R. También se incluye una somera descripción de la entrada y la salida producida al aplicarlo a uno de los ejemplos del texto, con el propósito de que esta información pueda ser útil para los que deseen desarrollar su propia herramienta de diseño.

El programa ha sido escrito en BASIC, y desarrollado y probado en un IBM PC XT de 256 K. Se compone de varias fases. Se ejecuta cargando la primera de ellas (con LOAD) y ejecutándola (RUN). A partir de aquí el programa va invocando a las fases siguientes conforme se van necesitando. Durante la ejecución se envían mensajes a la pantalla para indicar por dónde va avanzando el proceso. También se pueden tomar puntos de rearranque (checkpoints) para no perder todo el proceso realizado en caso de interrupciones voluntarias (teclas CTRL-Break) o involuntarias (fallo de potencia).

ENTRADA DE DATOS

Hay que describirle al programa el diagrama RE/R de partida, proporcionándole los nombres de todos los rectángulos, atributos y asociaciones, así como las cardinalidades de éstas.

Esta entrada de datos se hace a través de un diálogo conducido por el programa, respondiendo por pantalla a sus preguntas conforme las vaya haciendo. En ciertos momentos de este diálogo, el programa da la oportunidad de modificar y corregir los datos introducidos, por si se han tecleado errores.

El programa permite guardar en disco los datos de una sesión de trabajo, y completarlos o modificarlos en sesiones posteriores. Esto simplifica la entrada de

datos de diagramas complejos.También es útil para hacer simulaciones sobre cómo afectaría al diseño un cambio en una asociación, o en una cardinalidad, etc.

También se puede descomponer un diagrama en otros más pequeños, y procesar éstos independientemente. Luego se pueden fundir en otro más grande, introduciendo además nuevas interrelaciones entre ellos. Esto permite descomponer el diseño en varias etapas, o tener unos subdiagramas generales que formen parte de diversos diseños, evitando tener que teclear sus datos cada vez.

En las páginas siguientes se incluye el diálogo de pantalla para la introducción de la Base de Datos Universitaria cuyo diagrama RE/R se describió en el último ejemplo del capítulo anterior.

SALIDA

El diseño obtenido puede verse por pantalla, imprimirse o grabarse en disco. Esto último permitiría editarlo y modificarlo empleando un editor cualquiera del PC, o también enviarlo a un ordenador conectado al PC, donde podría ser utilizado para definir las tablas e índices si en este ordenador estuviera instalado el sistema relacional que vamos a usar.

La salida producida tiene cinco apartados:

a) *Entidades.* Listado de las entidades resultantes, indicando cuáles son fuertes, dependientes o de asociación.

b) *Tablas.* Listado de las tablas, sus atributos y sus claves.

c) *Asociaciones implícitas.* Listado de las asociaciones implícitas entre tablas, con sus cardinalidades.

d) *Condiciones de integridad.* Listado de las condiciones que se deducen de las cardinalidades especificadas en el diagrama.

e) *Protodefiniciones SQL.* Listado de las sentencias CREATE de SQL necesarias para definir las tablas y sus índices. Es útil si el sistema relacional que se usa emplea este lenguaje. Estas sentencias no incluyen las características de los atributos (por ejemplo, CHAR, VARCHAR, DECIMAL, etcécera.), por lo que hay que completarlas antes de introducirlas en el sistema. Tampoco incluyen la especificación de qué clave se designa como primaria ni de las reglas de mantenimiento de la Integridad Referencial (Restricción, Propagación o Anulación). Estas decisiones dependen de consideraciones semánticas no recogidas en el programa, por lo que hay que especificarlas manualmente.

Las sentencias de definición de índices se agrupan en dos tipos:

- Las que definen los índices correspondientes a las claves de las tablas.
- Las que definen índices que son convenientes para poder comprobar si se cumplen las condiciones de integridad de referencia deducidas del diagrama.

E

Cuando el programa aplica la regla de Agregación de Rectángulos, le asigna al rectángulo resultante un nombre que se compone concatenando los de los agregandos separados por un guión. Esto puede producir nombres arbitrarios poco significativos. Sin embargo, si el nombre de uno de los agregandos empieza por &, su nombre no interviene en la formación del nombre resultante. Por ello, en caso de que los nombres obtenidos sean poco significativos conviene modificar los del diagrama introduciendo otros con &. Así, por ejemplo, al aplicar el programa a la B. de D. Universitaria definida más arriba, se obtienen los nombres de entidades siguientes:

&2 (Entidad: Textos).
C1-C4 (Entidad: Asignaturas).
C10 (Entidad: Profesores).
C9-TUTOR (Entidad: Alumnos).
CALIFICACS (Entidad: Calificaciones).
GRUPOS-ENSEÑA (Entidad: Grupos).
MATRICS (Entidad: Matrículas).
PRERREQ (Entidad: Prerrequisitos).

Para obtener nombres más mnemotécnicos introducimos los cambios siguientes:

Al rectángulo C1 le llamamos &C1.
Al C4 le llamamos ASIGNATURAS.
Al C10, PROFESORES.
Al C9, ALUMNOS.
A la asociación entre C5 y C4 la llamamos TEXTOS.
A la asociación TUTOR la llamamos &TUTOR.
A la asociación ENSEÑA la llamamos &ENSEÑA.

En las páginas siguientes se incluye la salida producida después de introducir estos cambios de nombres.

APLICACION A LA B. D. UNIVERSITARIA

A continuación se incluye un listado del diálogo de pantalla para introducir los datos de la B. D. Universitaria del capítulo anterior. Este listado va seguido del diseño obtenido.

```
LOAD"dire10
Ok
RUN
***************  FASE 0  *****************
***************  FASE 1  *****************
¿Desea que se repitan las respuestas numeradas dadas en    -
    otra sesión? (S/N):n
¿Desea crear un fichero de respuestas con las de     -
    esta sesión? (S/N):n
(9)¿Desea arrancar en caliente (en el último checkpt)?: n
```

E

```
(1)¿Desea tracear datos? (S/N): n
(2)¿Desea tomar puntos de rearranque (checkpoints)? (S/N): s
(3)Nombre del fichero (si distinto del de por omisión):
(16)¿Desea fundir dos diagramas en uno? (S/N) : n
***************FIN DE FASE 1 *****************
********Empieza FASE 2 (Entrada de datos)******
**********Toma de datos*******************************
(4)¿Desea tomar los datos a partir de un fichero de entrada?: n
Título del diagrama:Base de Datos Universitaria
*****Introducir datos de RECTANGULOS:
Nombre de rectángulo (si no hay más dar intro):C1
     Nombre de atributo (si no hay más dar intro):NOMBRE
     Nombre de atributo (si no hay más dar intro):
Nombre de rectángulo (si no hay más dar intro):C2
     Nombre de atributo (si no hay más dar intro):#EXAM
     Nombre de atributo (si no hay más dar intro):
Nombre de rectángulo (si no hay más dar intro):C3
     Nombre de atributo (si no hay más dar intro):#GRUP
     Nombre de atributo (si no hay más dar intro):
Nombre de rectángulo (si no hay más dar intro):C4
     Nombre de atributo (si no hay más dar intro):#ASIG
     Nombre de atributo (si no hay más dar intro):
Nombre de rectángulo (si no hay más dar intro):C5
     Nombre de atributo (si no hay más dar intro):LIBRO
     Nombre de atributo (si no hay más dar intro):
Nombre de rectángulo (si no hay más dar intro):C6
     Nombre de atributo (si no hay más dar intro):DIA
     Nombre de atributo (si no hay más dar intro):
Nombre de rectángulo (si no hay más dar intro):C7
     Nombre de atributo (si no hay más dar intro):HORA
     Nombre de atributo (si no hay más dar intro):
Nombre de rectángulo (si no hay más dar intro):C8
     Nombre de atributo (si no hay más dar intro):AULA
     Nombre de atributo (si no hay más dar intro):
Nombre de rectángulo (si no hay más dar intro):C9
     Nombre de atributo (si no hay más dar intro):#ALUM
     Nombre de atributo (si no hay más dar intro):
Nombre de rectángulo (si no hay más dar intro):C10
     Nombre de atributo (si no hay más dar intro):#PROF
     Nombre de atributo (si no hay más dar intro):
Nombre de rectángulo (si no hay más dar intro):C11
     Nombre de atributo (si no hay más dar intro):NOMBRE
     Nombre de atributo (si no hay más dar intro):
Nombre de rectángulo (si no hay más dar intro):C12
     Nombre de atributo (si no hay más dar intro):SUELDO
     Nombre de atributo (si no hay más dar intro):
Nombre de rectángulo (si no hay más dar intro):C13
     Nombre de atributo (si no hay más dar intro):CATEG
     Nombre de atributo (si no hay más dar intro):
Nombre de rectángulo (si no hay más dar intro):C14
     Nombre de atributo (si no hay más dar intro):NOTA
     Nombre de atributo (si no hay más dar intro):
Nombre de rectángulo (si no hay más dar intro):C15
     Nombre de atributo (si no hay más dar intro):NOMBRE
     Nombre de atributo (si no hay más dar intro):
Nombre de rectángulo (si no hay más dar intro):C16
     Nombre de atributo (si no hay más dar intro):AÑO
     Nombre de atributo (si no hay más dar intro):
Nombre de rectángulo (si no hay más dar intro):PRERREQ
     Nombre de atributo (si no hay más dar intro):#ASIG
     Nombre de atributo (si no hay más dar intro):#ASIG
          Atr. repetido en el rectángulo, Sufijo=          2
```



```
    Nombre de atributo (si no hay más dar intro):
Nombre de rectángulo (si no hay más dar intro):GRUPOS
    Nombre de atributo (si no hay más dar intro):#ASIG
    Nombre de atributo (si no hay más dar intro):#GRUP
    Nombre de atributo (si no hay más dar intro):
Nombre de rectángulo (si no hay más dar intro):CLASES
    Nombre de atributo (si no hay más dar intro):#ASIG
    Nombre de atributo (si no hay más dar intro):#GRUP
    Nombre de atributo (si no hay más dar intro):DIA
    Nombre de atributo (si no hay más dar intro):
Nombre de rectángulo (si no hay más dar intro):CALIFICACS
    Nombre de atributo (si no hay más dar intro):#ASIG
    Nombre de atributo (si no hay más dar intro):#EXAM
    Nombre de atributo (si no hay más dar intro):#ALUM
    Nombre de atributo (si no hay más dar intro):
Nombre de rectángulo (si no hay más dar intro):
*****Introducir datos de CLAVES:
Nombre de rectángulo (o intro para acabar):
*****Introducir datos de ASOCIACIONES:
¿Desea definir una asociación (S / N)?:S
    Es una asociación explícita (S/N):? S
    Nombre de la asociación:&1
    Nombre de rectángulo  ORIGEN:C1
    Nombre de rectángulo DESTINO:C4
            Cardinalidad  origen:1
            Cardinalidad destino:1
¿Desea definir una asociación (S / N)?:S
    Es una asociación explícita (S/N):? S
    Nombre de la asociación:&2
    Nombre de rectángulo  ORIGEN:C5
    Nombre de rectángulo DESTINO:C4
            Cardinalidad  origen:M
            Cardinalidad destino:M
¿Desea definir una asociación (S / N)?:S
    Es una asociación explícita (S/N):? S
    Nombre de la asociación:&3
    Nombre de rectángulo  ORIGEN:C7
    Nombre de rectángulo DESTINO:CLASES
            Cardinalidad  origen:1
            Cardinalidad destino:M
¿Desea definir una asociación (S / N)?:S
    Es una asociación explícita (S/N):? S
    Nombre de la asociación:&4
    Nombre de rectángulo  ORIGEN:C8
    Nombre de rectángulo DESTINO:CLASES
            Cardinalidad  origen:1
            Cardinalidad destino:M
¿Desea definir una asociación (S / N)?:S
    Es una asociación explícita (S/N):? S
    Nombre de la asociación:&5
    Nombre de rectángulo  ORIGEN:C11
    Nombre de rectángulo DESTINO:C10
            Cardinalidad  origen:1
            Cardinalidad destino:M
¿Desea definir una asociación (S / N)?:S
    Es una asociación explícita (S/N):? S
    Nombre de la asociación:&6
    Nombre de rectángulo  ORIGEN:C13
    Nombre de rectángulo DESTINO:C10
            Cardinalidad  origen:1
            Cardinalidad destino:M
¿Desea definir una asociación (S / N)?:S
```

```
    Es una asociación explícita (S/N):? S
    Nombre de la asociación:&7
    Nombre de rectángulo  ORIGEN:C12
    Nombre de rectángulo DESTINO:C10
            Cardinalidad  origen:1
            Cardinalidad destino:M
¿Desea definir una asociación (S / N)?:S
    Es una asociación explícita (S/N):? S
    Nombre de la asociación:&8
    Nombre de rectángulo  ORIGEN:C16
    Nombre de rectángulo DESTINO:C9
            Cardinalidad  origen:1
            Cardinalidad destino:M
¿Desea definir una asociación (S / N)?:S
    Es una asociación explícita (S/N):? S
    Nombre de la asociación:&9
    Nombre de rectángulo  ORIGEN:C15
    Nombre de rectángulo DESTINO:C9
            Cardinalidad  origen:1
            Cardinalidad destino:M
¿Desea definir una asociación (S / N)?:S
    Es una asociación explícita (S/N):? S
    Nombre de la asociación:&10
    Nombre de rectángulo  ORIGEN:C14
    Nombre de rectángulo DESTINO:CALIFICACS
            Cardinalidad  origen:1
            Cardinalidad destino:M
¿Desea definir una asociación (S / N)?:S
    Es una asociación explícita (S/N):? S
    Nombre de la asociación:MATRICS
    Nombre de rectángulo  ORIGEN:C9
    Nombre de rectángulo DESTINO:GRUPOS
            Cardinalidad  origen:N
            Cardinalidad destino:M
¿Desea definir una asociación (S / N)?:S
    Es una asociación explícita (S/N):? S
    Nombre de la asociación:ENSEÑA
    Nombre de rectángulo  ORIGEN:C10
    Nombre de rectángulo DESTINO:GRUPOS
            Cardinalidad  origen:1
            Cardinalidad destino:N
¿Desea definir una asociación (S / N)?:S
    Es una asociación explícita (S/N):? S
    Nombre de la asociación:TUTOR
    Nombre de rectángulo  ORIGEN:C10
    Nombre de rectángulo DESTINO:C9
            Cardinalidad  origen:1
            Cardinalidad destino:N
¿Desea definir una asociación (S / N)?:S
    Es una asociación explícita (S/N):? N
    Nombre de rectángulo  ORIGEN:C4
    Nombre de rectángulo DESTINO:PRERREQ
            Cardinalidad  origen:1
            Cardinalidad destino:N
    Nombre de atr. común (si no hay más, intro):#ASIG
        Sufijo del atr. común en destino:1
    Nombre de atr. común (si no hay más, intro):
¿Desea definir una asociación (S / N)?:S
    Es una asociación explícita (S/N):? N
    Nombre de rectángulo  ORIGEN:C4
    Nombre de rectángulo DESTINO:PRERREQ
            Cardinalidad  origen:1
```

```
        Cardinalidad destino:N
    Nombre de atr. común (si no hay más, intro):#ASIG
        Sufijo del atr. común en destino:2
    Nombre de atr. común (si no hay más, intro):
¿Desea definir una asociación (S / N)?:S
    Es una asociación explícita (S/N):? N
    Nombre de rectángulo  ORIGEN:C2
    Nombre de rectángulo DESTINO:CALIFICACS
        Cardinalidad  origen:1
        Cardinalidad destino:M
    Nombre de atr. común (si no hay más, intro):#EXAM
    Nombre de atr. común (si no hay más, intro):
¿Desea definir una asociación (S / N)?:S
    Es una asociación explícita (S/N):? N
    Nombre de rectángulo  ORIGEN:C4
    Nombre de rectángulo DESTINO:CALIFICACS
        Cardinalidad  origen:1
        Cardinalidad destino:M
    Nombre de atr. común (si no hay más, intro):#ASIG
    Nombre de atr. común (si no hay más, intro):
¿Desea definir una asociación (S / N)?:S
    Es una asociación explícita (S/N):? N
    Nombre de rectángulo  ORIGEN:C4
    Nombre de rectángulo DESTINO:GRUPOS
        Cardinalidad  origen:1
        Cardinalidad destino:N
    Nombre de atr. común (si no hay más, intro):#ASIG
    Nombre de atr. común (si no hay más, intro):
¿Desea definir una asociación (S / N)?:S
    Es una asociación explícita (S/N):? N
    Nombre de rectángulo  ORIGEN:C3
    Nombre de rectángulo DESTINO:GRUPOS
        Cardinalidad  origen:1
        Cardinalidad destino:M
    Nombre de atr. común (si no hay más, intro):#GRUP
    Nombre de atr. común (si no hay más, intro):
¿Desea definir una asociación (S / N)?:S
    Es una asociación explícita (S/N):? N
    Nombre de rectángulo  ORIGEN:C6
    Nombre de rectángulo DESTINO:CLASES
        Cardinalidad  origen:1
        Cardinalidad destino:M
    Nombre de atr. común (si no hay más, intro):DIA
    Nombre de atr. común (si no hay más, intro):
¿Desea definir una asociación (S / N)?:S
    Es una asociación explícita (S/N):? N
    Nombre de rectángulo  ORIGEN:GRUPOS
    Nombre de rectángulo DESTINO:CLASES
        Cardinalidad  origen:1
        Cardinalidad destino:M
    Nombre de atr. común (si no hay más, intro):#ASIG
    Nombre de atr. común (si no hay más, intro):#GRUP
    Nombre de atr. común (si no hay más, intro):
¿Desea definir una asociación (S / N)?:S
    Es una asociación explícita (S/N):? N
    Nombre de rectángulo  ORIGEN:C9
    Nombre de rectángulo DESTINO:CALIFICACS
        Cardinalidad  origen:1
        Cardinalidad destino:M
    Nombre de atr. común (si no hay más, intro):#ALUM
    Nombre de atr. común (si no hay más, intro):
¿Desea definir una asociación (S / N)?:N
```

E

```
Cuando esté listo para seguir dé intro:
¿Desea modificar algún dato (S/N)?:N
¿Desea volver a introducir (o modificar) más datos:? N
(6)¿Desea salvar a disco los datos introducidos? (S/N): s
(7)Nombre del fichero de salida (incl. camino): bduniv.dat
Datos salvados
¿Desea listar los datos introducidos? (S/N):n
***************FIN DE FASE 2 *****************
****Empieza FASE 3 (Validación de datos) ****
Empieza clasificación de mclav por rect.-clave
Fin de clasifcn. de mclav por rect.
*****FIN DE VALIDACION DE DATOS *****
¿Desea reintroducir, modificar o salvar datos (S/N):? N
***************FIN DE FASE 3 *****************
**** Empieza FASE A (Listar datos de entrada)  ****
(12)¿Desea listar los datos de partida? (S/N): n
*************** FIN DE FASE A **************
**** Empieza FASE 4 (Definir claves)  ****
Empieza Definición de claves
Fin de Definir Claves
Empieza designación de claves primarias
Fin de designación de claves primarias
*****FIN DE APLICACION DE REGLA DE DEFINIR CLAVES *****
***************FIN DE FASE 4 *****************
*******EMPIEZA FASE 5 (Pasar asoc. expl. a impl.) ********
transformar asocs. explícitas en impl.
fin de transf. de asoc. expl. en impl.
*********** FIN DE FASE 5 ***************
**** Empieza FASE 6 (Aplicación de Reglas) ****
(Iteración:    1          )
Empieza Definición de claves
Nueva clave en rect.:        &1
Nueva clave en rect.:        &3
Nueva clave en rect.:        &4
Nueva clave en rect.:        &5
Nueva clave en rect.:        &6
(Cuando esté listo para seguir dé INTRO)
Nueva clave en rect.:        &7
Nueva clave en rect.:        &8
Nueva clave en rect.:        &9
Nueva clave en rect.:        &10
Nueva clave en rect.:        ENSEÑA
Nueva clave en rect.:        TUTOR
Nueva clave en rect.:        &1
Fin de Definir Claves
Empieza designación de claves primarias
En el rectángulo:            &1                ,la clave
     número                   2                ,pasa a ser primaria.
En el rectángulo:            &3                ,la clave
     número                   2                ,pasa a ser primaria.
En el rectángulo:            &4                ,la clave
     número                   2                ,pasa a ser primaria.
En el rectángulo:            &5                ,la clave
     número                   2                ,pasa a ser primaria.
En el rectángulo:            &6                ,la clave
     número                   2                ,pasa a ser primaria.
(Cuando esté listo para seguir dé INTRO)
En el rectángulo:            &7                ,la clave
     número                   2                ,pasa a ser primaria.
En el rectángulo:            &8                ,la clave
     número                   2                ,pasa a ser primaria.
En el rectángulo:            &9                ,la clave
```

E

```
        número              2                ,pasa a ser primaria.
En el rectángulo:          &10               ,la clave
        número              2                ,pasa a ser primaria.
En el rectángulo:          ENSEÑA            ,la clave
        número              2                ,pasa a ser primaria.
En el rectángulo:          TUTOR             ,la clave
        número              2                ,pasa a ser primaria.
Fin de designación de claves primarias
Invocar a la Fase 7 (Regla de Agregación de Rects.)
Empieza Agregación de Rects.
El rect.:     &1                ,
      ha sido agregado al: C4                ,
      cuyo nuevo nombre es:                  C4
El rect.:     &3                ,
      ha sido agregado al: CLASES            ,
      cuyo nuevo nombre es:                  CLASES
(Cuando esté listo para seguir dé INTRO)
El rect.:     &4                ,
      ha sido agregado al: CLASES            ,
      cuyo nuevo nombre es:                  CLASES
El rect.:     &5                ,
      ha sido agregado al: C10               ,
      cuyo nuevo nombre es:                  C10
El rect.:     &6                ,
      ha sido agregado al: C10               ,
      cuyo nuevo nombre es:                  C10
El rect.:     &7                ,
      ha sido agregado al: C10               ,
      cuyo nuevo nombre es:                  C10
El rect.:     &8                ,
      ha sido agregado al: C9                ,
      cuyo nuevo nombre es:                  C9
El rect.:     &9                ,
      ha sido agregado al: C9                ,
      cuyo nuevo nombre es:                  C9
(Cuando esté listo para seguir dé INTRO)
El rect.:     &10               ,
      ha sido agregado al: CALIFICACS        ,
      cuyo nuevo nombre es:                  CALIFICACS
El rect.:     ENSEÑA            ,
      ha sido agregado al: GRUPOS            ,
      cuyo nuevo nombre es:                  GRUPOS-ENSEÑA
El rect.:     TUTOR             ,
      ha sido agregado al: C9                ,
      cuyo nuevo nombre es:                  C9-TUTOR
El rect.:     C4                ,
      ha sido agregado al: C1                ,
      cuyo nuevo nombre es:                  C1-C4
Fin de Agregar Rectángulos
(17)¿Desea saltarse la Supresión de Rects.? (S/N): n
Empieza Supresión de Rects.
Suprimido el rectángulo:   C2
Suprimido el rectángulo:   C3
Suprimido el rectángulo:   C6
Suprimido el rectángulo:   C5
(Cuando esté listo para seguir dé INTRO)
Suprimido el rectángulo:   C7
Suprimido el rectángulo:   C8
Suprimido el rectángulo:   C11
Suprimido el rectángulo:   C13
Suprimido el rectángulo:   C12
Suprimido el rectángulo:   C16
```

```
Suprimido el rectángulo:    C15
Suprimido el rectángulo:    C14
Empieza Supresión de asocs. impl. (n:n)
*****Hay que hacer otra iteración  (INTRO):
(Iteración:     2          )
Empieza Definición de claves
Fin de Definir Claves
Empieza designación de claves primarias
Fin de designación de claves primarias
Empieza Agregación de Rects.
Fin de Agregar Rectángulos
Empieza Supresión de Rects.
Empieza Supresión de asocs. impl. (n:n)
**************FIN DE FASE 7 ****************
*****FIN DE APLICACION DE REGLAS (FASES 6 Y 7) *****
(Cuando esté listo para seguir dé INTRO)
**** Empieza FASE 8 (Eliminación claves redundantes) ****
Empieza clasificación de mreat por rect.
Fin de clasificación de mreat por rect.
inore:          1           nclav:          2
        Claves renumeradas en mtrab
inore:          9           nclav:          1
        Claves renumeradas en mtrab
inore:          10          nclav:          1
        Claves renumeradas en mtrab
inore:          17          nclav:          1
        Claves renumeradas en mtrab
inore:          18          nclav:          1
        Claves renumeradas en mtrab
inore:          19          nclav:          1
        Claves renumeradas en mtrab
inore:          20          nclav:          1
        Claves renumeradas en mtrab
inore:          22          nclav:          1
(Cuando esté listo para seguir dé INTRO)
        Claves renumeradas en mtrab
inore:          31          nclav:          1
        Claves renumeradas en mtrab
 Fin eliminación claves red.
*************** FIN DE FASE 8 *************
*********** FIN DE SUPR. CLAVES RED.*********
**** Empieza FASE 9 (Impresión del diseño obtenido)  ****
(15)¿Desea un informe del diagrama obtenido? (S/N): N
¿Desea guardar en disco el diagrama final obtenido?(S/N):n
*************** FIN DE FASE 9 *************
*********** FIN DE IMPRIMIR INFORME *********
Ok


Base de Datos Universitaria
===========================


I.- ENTIDADES.
______________


(El tipo de Entidad, entre paréntesis, significa:
    F =Fuerte ; D =Dependiente ; A =de asociación).
```

E

```
Ent. núm.  1 (D) : ALUMNOS
Ent. núm.  2 (F) : ASIGNATURAS
Ent. núm.  3 (A) : CALIFICACS
Ent. núm.  4 (D) : CLASES
Ent. núm.  5 (A) : GRUPOS
Ent. núm.  6 (A) : MATRICS
Ent. núm.  7 (A) : PRERREQ
Ent. núm.  8 (F) : PROFESORES
Ent. núm.  9 (D) : TEXTOS
```

II.- RELACIONES.

```
***REL. núm.  1: ALUMNOS

   ATRIBUTOS:
   ----------
      #ALUM
      AÑO
      NOMBRE
      #PROF

   CLAVE NUM. 1 :
   --------------
      #ALUM

***REL. núm.  2: ASIGNATURAS

   ATRIBUTOS:
   ----------
      NOMBRE
      #ASIG

   CLAVE NUM. 1 :
   --------------
      NOMBRE

   CLAVE NUM. 2 :
   --------------
      #ASIG

***REL. núm.  3: CALIFICACS

   ATRIBUTOS:
   ----------
      #ASIG
      #EXAM
      #ALUM
      NOTA

   CLAVE NUM. 1 :
   --------------
      #ASIG
      #EXAM
      #ALUM

***REL. núm.  4: CLASES
```

E

```
  ATRIBUTOS:
  ----------
    #ASIG
    #GRUP
    DIA
    HORA
    AULA

  CLAVE NUM. 1 :
  --------------
    #ASIG
    #GRUP
    DIA

***REL. núm.  5: GRUPOS

  ATRIBUTOS:
  ----------
    #ASIG
    #GRUP
    #PROF

  CLAVE NUM. 1 :

  --------------
    #ASIG
    #GRUP

***REL. núm.  6: MATRICS

  ATRIBUTOS:
  ----------
    #ALUM. 1
    #ASIG
    #GRUP

  CLAVE NUM. 1 :
  --------------
    #ALUM. 1
    #ASIG
    #GRUP

***REL. núm.  7: PRERREQ

  ATRIBUTOS:
  ----------
    #ASIG. 1
    #ASIG. 2

  CLAVE NUM. 1 :
  --------------
    #ASIG. 1
    #ASIG. 2

***REL. núm.  8: PROFESORES

  ATRIBUTOS:
  ----------
    #PROF
    NOMBRE
```

E

```
        CATEG
        SUELDO

     CLAVE NUM. 1 :
     --------------
        #PROF

***REL. núm.  9: TEXTOS

     ATRIBUTOS:
     ----------
        LIBRO
        #ASIG


       CLAVE NUM. 1 :
       --------------
          LIBRO
          #ASIG
```

III.- ASOCIACIONES IMPLICITAS.

```
  **ASOC. NUM.  2:

        Relcn. origen :ASIGNATURAS
        Relcn. destino:TEXTOS
        Cardinalidad  : (1 : m)

      Atributos comunes:
        1)Orig.:#ASIG
            Dest.:#ASIG

  **ASOC. NUM.  11:

        Relcn. origen :GRUPOS
        Relcn. destino:MATRICS
        Cardinalidad  : (1 : n)

      Atributos comunes:
        1)Orig.:#ASIG
            Dest.:#ASIG
        2)Orig.:#GRUP
            Dest.:#GRUP

  **ASOC. NUM.  14:

        Relcn. origen :ASIGNATURAS
        Relcn. destino:PRERREQ
        Cardinalidad  : (1 : n)

      Atributos comunes:
        1)Orig.:#ASIG
            Dest.:#ASIG. 1

  **ASOC. NUM.  15:

        Relcn. origen :ASIGNATURAS
        Relcn. destino:PRERREQ
        Cardinalidad  : (1 : n)
```

```
Atributos comunes:
  1)Orig.:#ASIG
    Dest.:#ASIG. 2

**ASOC. NUM.  17:

     Relcn. origen :ASIGNATURAS
     Relcn. destino:CALIFICACS
     Cardinalidad  : (1 : m)

   Atributos comunes:
     1)Orig.:#ASIG
       Dest.:#ASIG

**ASOC. NUM.  18:

     Relcn. origen :ASIGNATURAS
     Relcn. destino:GRUPOS
     Cardinalidad  : (1 : n)

   Atributos comunes:
     1)Orig.:#ASIG
       Dest.:#ASIG

**ASOC. NUM.  21:

     Relcn. origen :GRUPOS
     Relcn. destino:CLASES
     Cardinalidad  : (1 : m)

   Atributos comunes:
     1)Orig.:#ASIG
       Dest.:#ASIG
     2)Orig.:#GRUP
       Dest.:#GRUP

**ASOC. NUM.  22:

     Relcn. origen :ALUMNOS
     Relcn. destino:CALIFICACS
     Cardinalidad  : (1 : m)

   Atributos comunes:
     1)Orig.:#ALUM
       Dest.:#ALUM

**ASOC. NUM.  33:

     Relcn. origen :ALUMNOS

     Relcn. destino:MATRICS
     Cardinalidad  : (1 : m)

   Atributos comunes:
     1)Orig.:#ALUM
       Dest.:#ALUM. 1
```

E

```
**ASOC. NUM.  34:

     Relcn. origen :PROFESORES
     Relcn. destino:GRUPOS
     Cardinalidad  : (1 : n)

   Atributos comunes:
     1)Orig.:#PROF
        Dest.:#PROF

**ASOC. NUM.  35:

     Relcn. origen :PROFESORES
     Relcn. destino:ALUMNOS
     Cardinalidad  : (1 : n)

   Atributos comunes:
     1)Orig.:#PROF
        Dest.:#PROF
```

IV.- CONDICIONES DE INTEGRIDAD.

```
** 1).Los valores que toman la, o las, columnas:
           #ASIG
      en las filas de la tabla:
            TEXTOS
      tienen que existir, y SIN repetirse,
      en las filas de la tabla:
            ASIGNATURAS
      en la, o las, columnas:
          #ASIG
      respectivamente.

** 2).Los valores que toman la, o las, columnas:
           #ASIG
      en las filas de la tabla:
            ASIGNATURAS
      tienen que existir, pudiendo REPETIRSE,
      en las filas de la tabla:


            TEXTOS
      en la, o las, columnas:
          #ASIG
      respectivamente.

** 3).Los valores que toman la, o las, columnas:
          #ASIG
          #GRUP
      en las filas de la tabla:
            MATRICS
      tienen que existir, y SIN repetirse,
      en las filas de la tabla:
            GRUPOS
      en la, o las, columnas:
          #ASIG
          #GRUP
      respectivamente.
```

E

```
** 4).Los valores que toman la, o las, columnas:
        #ASIG. 1
     en las filas de la tabla:
           PRERREQ
     tienen que existir, y SIN repetirse,
     en las filas de la tabla:
           ASIGNATURAS
     en la, o las, columnas:
        #ASIG
     respectivamente.

** 5).Los valores que toman la, o las, columnas:
        #ASIG. 2
     en las filas de la tabla:
           PRERREQ
     tienen que existir, y SIN repetirse,
     en las filas de la tabla:
           ASIGNATURAS
     en la, o las, columnas:
        #ASIG
     respectivamente.

** 6).Los valores que toman la, o las, columnas:
        #ASIG
     en las filas de la tabla:
           CALIFICACS
     tienen que existir, y SIN repetirse,
     en las filas de la tabla:
           ASIGNATURAS
     en la, o las, columnas:
        #ASIG
     respectivamente.

** 7).Los valores que toman la, o las, columnas:
        #ASIG
     en las filas de la tabla:
           ASIGNATURAS
     tienen que existir, pudiendo REPETIRSE,
     en las filas de la tabla:
           CALIFICACS
     en la, o las, columnas:
        #ASIG
     respectivamente.

** 8).Los valores que toman la, o las, columnas:
        #ASIG
     en las filas de la tabla:
           GRUPOS
     tienen que existir, y SIN repetirse,
     en las filas de la tabla:
           ASIGNATURAS
     en la, o las, columnas:
        #ASIG
     respectivamente.

** 9).Los valores que toman la, o las, columnas:
        #ASIG
        #GRUP
```

```
        en las filas de la tabla:
             CLASES
        tienen que existir, y SIN repetirse,
        en las filas de la tabla:
             GRUPOS
        en la, o las, columnas:
           #ASIG
           #GRUP
        respectivamente.

** 10).Los valores que toman la, o las, columnas:
           #ASIG
           #GRUP
        en las filas de la tabla:
             GRUPOS
        tienen que existir, pudiendo REPETIRSE,
        en las filas de la tabla:
             CLASES
        en la, o las, columnas:
           #ASIG
           #GRUP
        respectivamente.

** 11).Los valores que toman la, o las, columnas:
           #ALUM
        en las filas de la tabla:
             CALIFICACS
        tienen que existir, y SIN repetirse,
        en las filas de la tabla:
             ALUMNOS
        en la, o las, columnas:
           #ALUM
        respectivamente.

** 12).Los valores que toman la, o las, columnas:
           #ALUM
        en las filas de la tabla:
             ALUMNOS
        tienen que existir, pudiendo REPETIRSE,
        en las filas de la tabla:
             CALIFICACS
        en la, o las, columnas:
           #ALUM
        respectivamente.

** 13).Los valores que toman la, o las, columnas:
           #ALUM. 1
        en las filas de la tabla:
             MATRICS
        tienen que existir, y SIN repetirse,
        en las filas de la tabla:
             ALUMNOS
        en la, o las, columnas:
           #ALUM
        respectivamente.
```

E

```
** 14).Los valores que toman la, o las, columnas:
        #ALUM
     en las filas de la tabla:
           ALUMNOS
     tienen que existir, pudiendo REPETIRSE,
     en las filas de la tabla:
           MATRICS
     en la, o las, columnas:
        #ALUM. 1
     respectivamente.


** 15).Los valores que toman la, o las, columnas:
        #PROF
     en las filas de la tabla:
           GRUPOS
     tienen que existir, y SIN repetirse,
     en las filas de la tabla:
           PROFESORES
     en la, o las, columnas:
        #PROF
     respectivamente.


** 16).Los valores que toman la, o las, columnas:
        #PROF
     en las filas de la tabla:
           ALUMNOS
     tienen que existir, y SIN repetirse,
     en las filas de la tabla:
           PROFESORES
     en la, o las, columnas:
        #PROF
     respectivamente.
```

V.- PROTODEFINICIONES SQL.

```
**** TABLAS E INDICES DE CLAVES:

  CREATE TABLE ALUMNOS
     (#ALUM NOT NULL
     ,AÑO NOT NULL
     ,NOMBRE NOT NULL
     ,#PROF NOT NULL
     )

    CREATE UNIQUE INDEX ALUMNOX1
        ON ALUMNOS
        (#ALUM
        )

  CREATE TABLE ASIGNATURAS
     (NOMBRE NOT NULL
     ,#ASIG NOT NULL
     )
```

E

```
CREATE UNIQUE INDEX ASIGNAX1
    ON ASIGNATURAS
    (NOMBRE
    )

CREATE UNIQUE INDEX ASIGNAX2
    ON ASIGNATURAS

      (#ASIG
      )

CREATE TABLE CALIFICACS
    (#ASIG NOT NULL
    ,#EXAM NOT NULL
    ,#ALUM NOT NULL
    ,NOTA NOT NULL
    )

  CREATE UNIQUE INDEX CALIFIX1
      ON CALIFICACS
      (#ASIG
      ,#EXAM
      ,#ALUM
      )

CREATE TABLE CLASES
    (#ASIG NOT NULL
    ,#GRUP NOT NULL
    ,DIA NOT NULL
    ,HORA NOT NULL
    ,AULA NOT NULL
    )

  CREATE UNIQUE INDEX CLASESX1
      ON CLASES
      (#ASIG
      ,#GRUP
      ,DIA
      )

CREATE TABLE GRUPOS
    (#ASIG NOT NULL
    ,#GRUP NOT NULL
    ,#PROF NOT NULL
    )

  CREATE UNIQUE INDEX GRUPOSX1
      ON GRUPOS
      (#ASIG
      ,#GRUP
      )

CREATE TABLE MATRICS
    (#ALUM_1 NOT NULL
    ,#ASIG NOT NULL
    ,#GRUP NOT NULL
    )
```

```
CREATE UNIQUE INDEX MATRICX1
    ON MATRICS
    (#ALUM_1
    ,#ASIG
    ,#GRUP
    )


CREATE TABLE PRERREQ
  (#ASIG_1 NOT NULL
  ,#ASIG_2 NOT NULL
  )


CREATE UNIQUE INDEX PRERREX1
    ON PRERREQ
    (#ASIG_1
    ,#ASIG_2
    )


CREATE TABLE PROFESORES
  (#PROF NOT NULL
  ,NOMBRE NOT NULL
  ,CATEG NOT NULL
  ,SUELDO NOT NULL
  )


CREATE UNIQUE INDEX PROFESX1
    ON PROFESORES
    (#PROF
    )


CREATE TABLE TEXTOS
  (LIBRO NOT NULL
  ,#ASIG NOT NULL
  )


CREATE UNIQUE INDEX TEXTOSX1
    ON TEXTOS
    (LIBRO
    ,#ASIG
    )

**** INDICES PARA CONDS. DE INTEGRIDAD:

CREATE INDEX TEXTOSY2
    ON TEXTOS
    (#ASIG
    )
CREATE INDEX CALIFIY17

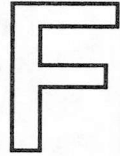

    ON CALIFICACS
    (#ASIG
    )
CREATE INDEX CLASESY21
    ON CLASES
```

E

```
     (#ASIG
     ,#GRUP
     )
CREATE INDEX CALIFIY22
     ON CALIFICACS
     (#ALUM
     )
CREATE INDEX MATRICY33
     ON MATRICS
     (#ALUM_1
     )


************* FIN DEL INFORME ***************
```

APENDICE



Soluciones a los ejercicios propuestos

ALGEBRA RELACIONAL

1) *Sean las relaciones T y S siguientes:*

R	A	B
	a	b
	c	b
	d	e

S	B	C
	b	c
	b	d
	e	a

Hallar los resultados de las siguientes expresiones:

1.1) R ∪ S.

	a	b
	c	b
	d	e
	b	c
	b	d
	e	a

1.2) R – S.

	a	b
	c	b
	d	e

F

1.3) R * S.

A	B	C
a	b	c
a	b	d
c	b	c
c	b	d
d	e	a

1.4) **P**(A) (R).

A
a
c
d

1.5) **S**(A = C) (R × S).

A	B	B	C
a	b	e	a
c	b	b	c
d	e	b	d

2) *Sean las relaciones siguientes:*

a) HOMBRES (NOMH, EDAD)
Clave: (NOMH).
Significado: Cada fila representa un hombre, cuyo nombre es NOMH, y su edad en años viene en EDAD.

b) MUJERES (NOMM, EDAD)
Clave: (NOMM).
Significado: Cada fila representa una mujer, cuyo nombre es NOMM, y su edad en años viene en EDAD.

c) HSIM (NOMH, NOMM).
Clave: (NOMH, NOMM).
Significado: El hombre NOMH cae simpático a la mujer NOMM.

d) MSIM (NOMH, NOMM).
Clave: (NOMH, NOMM).
Significado: La mujer NOMM cae simpática al hombre NOMH.

e) MATRIM (NOMH, NOMM).
Clave: (NOMH, NOMM).
Significado: La pareja NOMH y NOMM están casados.

F

Escribir las sentencias necesarias para responder a las preguntas siguientes:

2.1) Hallar las parejas de hombres y mujeres que se caen mutuamente simpáticos.
HSIM ∩ MSIM.

2.2) Hallar las parejas de hombres y mujeres que se caen mutuamente simpáticos, con edades entre 20 y 30 años y que no estén casados entre sí.
(**S**((20 ⩽ HOMBRES.EDAD ⩽ 30) ∧ (20 ⩽ MUJERES.EDAD ⩽ 30)) ((HSIM ∩ MSIM) * HOMBRES * MUJERES)) – MATRIM.

2.3) Hallar los matrimonios en que ambos esposos se caen mutuamente simpáticos.
HSIM ∩ MSIM ∩ MATRIM.

2.4) Hallar las mujeres casadas a quienes no cae simpático su marido.
P(NOMM) (MATRIM – HSIM).

2.5) Hallar los hombres misóginos a quienes no cae simpática ninguna mujer.
P(NOMH) (HOMBRES) – **P**(NOMH) (MSIM).

2.6) Hallar los hombres y mujeres asociales a quienes no cae nadie simpático.
(**P**(NOMH) (HOMBRES) – **P**(NOMH) (MSIM)) ∪ (**P**(NOMM) (MUJERES) – **P**(NOMM) (HSIM)).

2.7) Hallar las mujeres casadas que caen simpáticas a algún hombre.
P(NOMM) (MSIM) – **P**(NOMM) (MATRIM).

2.8) Hallar los hombres a quienes sólo caen simpáticas mujeres casadas.

Sea MSIMNC (NOMH, NOMM) el conjunto de parejas de hombres con mujeres simpáticas no casadas. Será igual a:
MSIMNC(NOMH, NOMM) = MSIM – (MSIM * **P**(NOMM) (MATRIM)).

La respuesta a la pregunta será:
P(NOMH) (MSIM) – **P**(NOMH) (MSIMNC).

Expresada en una sola fórmula queda:
P(NOMH) (MSIM) – **P**(NOMH) (MSIM – (MSIM * **P**(NOMM) (MATRIM)).

3) *Sean las relaciones siguientes:*

a) SOCIO (AFICIONADO, VIDEOCLUB).
Significado: AFICIONADO es SOCIO de VIDEOCLUB.

b) GUSTA (AFICIONADO, PELICULA).
Significado: PELICULA de cine GUSTA a AFICIONADO.

c) VIDEOTECA (VIDEOCLUB, PELICULA).
Significado: VIDEOCLUB dispone en su VIDEOTECA de PELICULA.

Escribir las sentencias necesarias para responder a las preguntas siguientes:

3.1) Videoclubes que disponen de alguna película que le guste a José Pérez.
P(VIDEOCLUB) (**S**(AFICIONADO = JOSE PEREZ) (VIDEOTECA * GUSTA)).

3.2) Aficionados que son socios al menos de un videoclub que dispone de alguna película de su gusto.
P(AFICIONADO) (SOCIO * VIDEOTECA * GUSTA).

F

3.3) Aficionados que son socios solamente de videoclubes que disponen de alguna película de su gusto.
P(AFICIONADO) (SOCIO) – P(AFICIONADO) (SOCIO – P(AFICIONADO, VIDEOCLUB) (SOCIO * VIDEOTECA * GUSTA)).

3.4) Aficionados que no son socios de nigún videoclub donde tengan alguna película de su gusto.
P(AFICIONADO) (SOCIO) – P(AFICIONADO) (SOCIO * VIDEOTECA * GUSTA).

4) *Sean las relaciones siguientes:*

a) PRO (NP, NOMP, CIUDADP).
Clave: (NP).
Significado: Cada fila representa un Proveedor, cuyo identificador es NP, su nombre NOMP, y habita en la ciudad CIUDADP.

b) ART (NA, DESA, COLOR, TALLA).
Clave: (NA).
Significado: Cada fila representa un Artículo, cuyo identificador es NA y su descripción DESA.

c) FAB (NF, NOMF, CIUDADF).
Clave: (NF).
Significado: Cada fila representa una Fábrica, cuyo identificador es NF, su nombre NOMF, y está situada en CIUDADF.

d) PED (NP, NA, NF, CANTIDAD).
Clave: (NP, NA, NF).
Significado: Cada fila representa un Pedido del Artículo NA, al Proveedor NP, para la Fábrica NF.

Escribir las sentencias necesarias para responder a las preguntas siguientes:

4.1) Hallar los identificadores de todas las Fábricas.
P(NF) (FAB).

4.2) Hallar los nombre de las Fábricas situadas en Madrid.
P(NOMF) (S(CIUDADF = MADRID) (FAB)).

4.3) Artículos tales que ningún otro tiene talla más pequeña.
Sean ART1 = ART y ART2 = ART (vamos a combinar las filas de ART consigo mismas):
P(NA) (ART) – P(ART1.NA) (S(ART1.TALLA > ART2.TALLA) (ART1 × ART2))

4.4) Proveedores que suministran a la Fábrica F1.
P(NP) (S(NF = F1) (PED)).

4.5) Proveedores que suministran a la Fábrica F1 el Artículo A1.
P(NP) (S((NF = F1) ∧ (NA = A1)) (PED)).

4.6) Nombres de las Fábricas a las que suministra el Proveedor P1.
P(NOMF) (S(NP = P1) (PED) * FAB)).

4.7) Colores de los Artículos suministrados por el Proveedor P1.
P(COLOR) (S(NP = P1) (PED * FAB)).

4.8) Qué Proveedores suministran a las Fábricas F1 y F2.
Sean PED1 = PED y PED2 = PED:
P(NP) (**S**((PED1.NP = PED2.NP) ∧ (PED1.NF = F1) ∧ (PED2.NF = F2)) (PED1 × PED2)).
O también:
(**P**(NP) (**S**(NF = F1) (PED))) ∩ (**P**(NP) (**S**(NF = F2) (PED))).

4.9) Proveedores que suminstran Artículos azules a la Fábrica F1.
P(NP) (**S**((NF = F1) ∧ (COLOR = Azul)) (PED * ART)).

4.10) Artículos suministrados a las Fábricas de Madrid.
P(NA) (**S**(CIUDAD = Madrid) (PED * FAB)).

4.11) Proveedores que suministran algún Artículo azul a las Fábricas de Madrid o Lisboa.
P(NP) (**S**(((CIUDAD = Madrid) ∨ (CIUDAD = Lisboa)) ∧ (COLOR = Azul)) (PED * FAB * ART)).

4.12) Artículos suministrados por Proveedores en cuya ciudad hay alguna Fábrica.
P(NA) (PED * (**P**(NP) (**S**(CIUDADP = CIUDADF) (PRO * FAB)))).

4.13) Artículos suministrados a las Fábricas de Madrid por Proveedores de Madrid.
P(NA) (**S**((CIUDADP = Madrid) ∧ (CIUDADF = Madrid)) (PED * PRO * FAB))).

4.14) Fábricas abastecidas por al menos un Proveedor de distinta ciudad.
P(NF) (**S**(CIUDADP ¬ = CIUDADF) (PED * PRO * FAB)).

4.15) Fábricas que no son abastecidas de Artículos azules por Proveedores de Madrid.
P(NF) (FAB) − **P**(NF) (**S**((CIUDADP = Madrid) ∧ (COLOR = Azul)) (PED * PRO * ART)).

4.16) Proveedores que suministran al menos un Artículo suministrado por al menos otro Proveedor que suministra al menos un Artículo azul.

Sea R1 = (Proveedores de Artículos azules):
R1 (NP) = **P**(NP) (**S**(COLOR = Azul) (PED * ART)).

Sea R2 = (Artículos suministrados por los Proveedores de Artículos azules):
R2 (NA) = **P**(NA) (PED * R1).

Sea R3 = (Proveedores de los Artículos de R2):
R3 (NP) = **P**(NP) (PED * R2).

En conclusión la fórmula que responde a la pregunta es:
P(NP) (PED * **P**(NA) (PED * **P**(NP) (**S**(COLOR = Azul) (PED * ART)))).

4.17) Fábricas que usan al menos un Artículo suministrado por el Proveedor P1.
P(NF) (PED * (**P**(NA) (**S**(NP = P1) (PED)))).

4.18) Parejas de ciudades tales que un Proveedor de la primera abastece a una Fábrica de la segunda.
P(CIUDADP, CIUDADF) (PED * PRO * FAB).

4.19) Obtener las tripletas de valores de ⟨CIUDAD, NA, CIUDAD⟩ tales que un Proveedor de la primera ciudad abastece el artículo NA a una Fábrica de la segunda ciudad.
P(CIUDADP, NA, CIUDADF) (PED * PRO * FAB).

F

4.20) Repetir la pregunta anterior, pero sin obtener las tripletas en que ambas ciudades son la misma.
S(CIUDADP¬ =CIUDADF) (P(CIUDADP, NA, CIUDADF) (PED * PRO * FAB)).

4.21) Proveedores que suministran un mismo Artículo, al menos, a todas las Fábricas.
P(NP) ((**P**(NP, NA, NF) (PED))/(**P**(NF) (FAB))).

4.22) Fábricas que tienen como único Proveedor a P1.
P(NF) (**S**(NP=P1) (PED)) − **P**(NF) (**S**(NP¬ =P1) (PED)).

4.23) Artículos que son suministrados a todas las Fábricas de Madrid (excluyendo los que sólo se suministran a algunas).
(**P**(NA, NF) (PED))/(**P**(NF) (**S**(CIUDADF=Madrid) (FAB))).

4.24) Fábricas que usan, al menos, todos los Artículos suministrados por el Proveedor P1.
(**P**(NF, NA) (PED))/(**P**(NA) (**S**(NP=P1) (PED))).

4.25) Fábricas que usan sólo Artículos que pueden ser suministrados por el Proveedor P1.
(**P**(NF) (FAB)) − (**P**(NF) (PED * ((**P**(NA) (ART)) − **P**(NP) (**S**(NP=P1) (PED))))).

4.26) Fábricas abastecidas por el Proveedor P1 con todos los Artículos que éste suministra.
(**P**(NP, NA, NF) (PED))/(**P**(NP, NA) (**S**(NP=P1) (PED))).

4.27) Fábricas que obtienen del Proveedor P1, total o parcialmente, todos los Artículos que usan.
(**P**(NF) (FAB)) − (**P**(NF) ((**P**(NA, NF) (PED)) − (**P**(NA, NF) (**S**(NP=P1) (PED))))).

4.28) Fábricas abastecidas por todos los Proveedores que suministran algún Artículo de color azul.
(**P**(NF, NP) (PED))/(**P**(NP) (**S**(COLOR=Azul) (PED * ART))).

5) *Sean las relaciones siguientes:*

a) EMPM (#E, #D, NOME, CAT, SUELDO).
Hay una tupla por cada empleado destinado en Madrid.

#E: Número de empleado.
#D: Número de departamento en el que trabaja.
NOME: Nombre del empleado.
CAT: Categoría del empleado.
SUELDO: Sueldo del empleado.

Clave Primaria: #E.

b) EMPL (#E, #D, NOME, CAT, SUELDO).

Hay una tupla por cada empleado destinado en Lisboa.
Iguales atributos que EMPM.
Todos los empleados de la empresa están, o bien en EMPM, o bien en EMPL. Un empleado no pude estar a la vez en EMPM y EMPL.
Clave Primaria: #E.

F

c) DEP (#D, NOMD, JEFE, ORG, LOCAL).

Hay una tupla por cada departamento de la empresa, sea cual sea la ciudad donde se encuentre (Madrid o Lisboa).

#D: Número de departamento.

NOMD: Nombre del departamento.

JEFE: Número de empleado del director del departamento.

ORG: Número de departamento del que este departamento depende. Se supone que en el organigrama de la empresa cada departamento depende de otro, y solamente de él, en una estructura jerárquica. Naturalmente habría de existir un departamento origen, que no depende de ningún otro.

LOCAL: Dirección (es decir, ciudad, calle, número, planta) donde se encuentra ubicada la oficina del departamento. Algunos de estos locales son propiedad de la empresa y otros están alquilados.

Claves: #D, NOMD, JEFE.

Clave Primaria: #D.

d) ALQ (LOCAL, CANTIDAD).

Hay una tupla por cada local alquilado. Cada local debe contener al menos un departamento.

LOCAL: El mismo significado que en DEP.

CANTIDAD: Coste del alquiler del local.

Clave Primaria: LOCAL.

Preguntas:

a) ¿Qué atributos no deben de tomar valores nulos de acuerdo con la regla de Integridad de Claves? (Integridad de Entidad).

b) ¿Qué atributos son Claves Ajenas? (Integridad de Referencia).

Pregunta *a*)
Todos los atributos designados como Claves Primarias: EMPM.#E, EMPL.#E, DEP.#D y ALQ.LOCAL.

Pregunta *b*)
Todos los atributos cuyos valores no nulos deban de existir en la Clave Primaria de alguna relación, que son:

- EMPM.#D (hace referencia a la Clave Primaria DEP.#D).
- EMPL.#D (hace referencia a la Clave Primaria DEP.#D).
- JEFE (hace referencia, o bien a la Clave Primaria EMPM.#E, o bien a la EMPL.#E).
- ORG (hace referencia a la Clave Primaria DEP.#D).

El atributo DEP.LOCAL no es una clave ajena aunque LOCAL sea Clave Primaria en ALQ, pues no todos los valores de DEP.LOCAL han de estar en ALQ.LOCAL. En cambio, todos los locales alquilados han de contener algún Departamento, por lo que todos los valores de ALQ.LOCAL han de existir en DEP.LOCAL. Sin embargo, esta condición de existencia no es una condición de Integridad de Referencia, en el sentido estricto en que suelen definirse este tipo de condiciones, pues el atributo referido, DEP.LOCAL, no es una clave primaria.

F

CALCULO RELACIONAL

1) *Sean las relaciones siguientes:*

a) HOMBRES (NOMH, EDAD).
Clave: (NOMH). Significado: Cada fila representa un hombre, cuyo nombre es NOMH y su edad en años viene en EDAD.

b) MUJERES (NOMM, EDAD).
Clave: (NOMM).
Significado: Cada fila representa una mujer, cuyo nombre es NOMM y su edad en años viene en EDAD.

c) HSIM (NOMH, NOMM).
Clave: (NOMH, NOMM).
Significado: El hombre NOMH cae simpático a la mujer NOMM.

d) MSIM (NOMH, NOMM).
Clave: (NOMH, NOMM).
Significado: La mujer NOMM cae simpática al hombre NOMH.

e) MATRIM (NOMH, NOMM).
Clave: (NOMH, NOMM).
Significado: La pareja NOMH y NOMM están casados.

Escribir las sentencias necesarias para responder a las preguntas siguientes.

En las fórmulas que siguen, h, m, c, hs y ms son tuplas variables que toman valores en las relaciones:

$DOM(h) = HOMBRES,\ DOM(m) = MUJERES,\ DOM(c) = MATRIM,$
$DOM(hs) = HSIM,\ DOM(ms) = MSIM.$

1.1) Hallar las parejas de hombres y mujeres que se caen mutuamente simpáticos.
$t \mid HSIM(t) \wedge MSIM(t).$

1.2) Hallar las parejas de hombres y mujeres que se caen mutuamente simpáticos, con edades entre 20 y 30 años y que no estén casados entre sí.
$t \mid HSIM(t) \wedge MSIM(t) \wedge (\neg MATRIM(t)) \wedge$
$(\exists h((20 \leqslant h.EDAD \leqslant 30) \wedge (h.NOMH = t.NOMH))) \wedge$
$(\exists m((20 \leqslant m.EDAD \leqslant 30) \wedge (m.NOMH = t.NOMM))).$

1.3) Hallar los matrimonios en que ambos esposos se caen mutuamente simpáticos.
$t \mid HSIM(t) \wedge MSIM(t) \wedge MATRIM(t).$

1.4) Hallar las mujeres casadas a quienes no cae simpático su marido.
$t_{(1} \mid \exists c((c.NOMM = t) \wedge (\forall hs(c \neg = hs))).$

1.5) Hallar los hombres misóginos a quienes no cae simpática ninguna mujer.
$t_{(1} \mid (\exists h(h.NOMH = t)) \wedge (\forall ms(ms.NOMH \neg = t)).$

1.6) Hallar los hombres y mujeres asociales a quienes no cae nadie simpático.
$t_{(1} \mid ((\exists h(h.NOMH = t)) \wedge (\forall ms(ms.NOMH \neg t))) \vee ((\exists m(m.NOMM = t))$
$\wedge (\forall hs(hs.NOMM \neg = t)).$

1.7) Hallar las mujeres casadas que caen simpáticas a algún hombre.
$t_{(1} \mid (\exists c(c.NOMM = t)) \wedge (\exists ms(ms.NOMM = t)).$

1.8) Hallar los hombres a quienes sólo caen simpáticos mujeres casadas.

$t_{(1} \mid (\exists h(h.NOMH = t)) \wedge (\forall ms((ms.NOMH\neg = t) \vee ((ms.NOMH = t) \wedge (\exists m(m.NOMM = ms.NOMM))))).$

2) *Sean las relaciones siguientes:*

a) SOCIO (AFICIONADO, VIDEOCLUB).
Significado: AFICIONADO es SOCIO de VIDEOCLUB.

b) GUSTA (AFICIONADO, PELICULA).
Significado: PELICULA de cine GUSTA a AFICIONADO.

c) VIDEOTECA (VIDEOCLUB, PELICULA).
Significación: VIDEOCLUB dispone en su VIDEOTECA de PELICULA.

Escribir las sentencias necesarias para responder a las preguntas siguientes.

En las fórmulas que siguen, s, g y v son variables que toman valores en los dominios:
DOM(s) = SOCIO, DOM(g) = GUSTA, DOM(v) = VIDEOTECA.

2.1) Videoclubes que disponen de alguna película que le guste a José Pérez.

$t_{(1} \mid \exists g\, \exists v((g.AFICIONADO = \text{José Pérez}) \wedge (g.PELICULA = v.PELICULA) \wedge (v.VIDEOCLUB = t)).$

2.2) Aficionados que son socios al menos de un videoclub que dispone de alguna película de su gusto.

$t_{(1} \mid \exists s\, \exists v\, \exists g((s.VIDEOCLUB = v.VIDEOCLUB) \wedge (v.PELICULA = g.PELICULA) \wedge (g.AFICIONADO = s.AFICIONADO)).$

2.3) Aficionados que son socios solamente de videoclubes que disponen de alguna película de su gusto.

$t_{(1} \mid \forall s((s.AFICIONADO\neg = t) \vee ((s.AFICIONADO = t) \wedge (\exists v\, \exists g((s.VIDEOCLUB = v.VIDEOCLUB) \wedge (v.PELICULA = g.PELICULA) \wedge (g.AFICIONADO = t))))).$

2.4) Aficionados que no son socios de ningún videoclub donde tengan alguna película de su gusto.

$t_{(1} \mid \forall s((s.AFICIONADO\neg = t) \vee ((s.AFICIONADO = t) \wedge \neg(\exists v\, \exists g((s.VIDEOCLUB = v.VIDEOCLUB) \wedge (v.PELICULA = g.PELICULA) \wedge (g.AFICIONADO = t))))).$

3) *Sean las relaciones siguientes:*

a) PRO (NP, NOMP, CIUDADP).
Clave: (NP).
Significado: Cada fila representa un Proveedor, cuyo identificador es NP, su nombre NOMP, y habita en la ciudad CIUDADP.

b) ART (NA, DESA, COLOR, TALLA).
Clave: (NA).
Significado: Cada fila representa un Artículo, cuyo identificador es NA y su descripción DESA.

c) FAB (NF, NOMF, CIUDADF).
Clave: (NF).
Significado: Cada fila representa una Fábrica, cuyo identificador es NF, su nombre NOMF, y está situada en CIUDADF.

F

d) PED (NP, NA, NF, CANTIDAD).
Clave: (NP, NA, NF).
Significado: Cada fila representa un Pedido del Artículo NA, al Proveedor NP, para la Fábrica NF.

Escribir las sentencias necesarias para responder a las preguntas siguientes.

En las fórmulas que siguen, p, a, f, x, y, z, son variables que toman valores en los dominios:

$DOM(p)=PRO,\ DOM(a)=ART,\ DOM(f)=FAB,\ DOM(x)=DOM(y)=DOM(z)=PED.$

3.1) Hallar los identificadores de todas las Fábricas.
$t_{(1} \mid \exists f(f.NF=t).$

3.2) Hallar los nombres de las Fábricas situadas en Madrid.
$t_{(1} \mid \exists f((f.NOMF=t) \wedge (f.CIUDADF=Madrid)).$

3.3) Artículos tales que ningún otro tiene talla más pequeña.
Para $DOM(r)=ART$:
$t_{(1} \mid \exists a(a.NA=t \wedge \forall r(a.TALLA \leqslant r.TALLA)).$

3.4) Proveedores que suministran a la Fábrica F1.
$t_{(1} \mid \exists x(x.NP=t \wedge x.NF=F1).$

3.5) Proveedores que suministran a la Fábrica F1 el Artículo A1.
$t_{(1} \mid \exists x(x.NP=t \wedge x.NF=F1 \wedge x.NA=A1).$

3.6) Nombres de las Fábricas a las que suministra el Proveedor P1.
$t_{(1} \mid \exists x\, \exists f(x.NP=P1 \wedge x.NF=f.NF \wedge f.NOMF=t)).$

3.7) Colores de los Artículos suministrados por el Proveedor P1.
$t_{(1} \mid \exists x\, \exists a(x.NP=P1 \wedge x.NA=a.NA \wedge a.COLOR=t)).$

3.8) Qué Proveedores suministran a las Fábricas F1 y F2.
$t_{(1} \mid (\exists x(x.NF=F1 \wedge x.NP=t)) \wedge (\exists x(x.NF=F2 \wedge x.NP=t)).$

3.9) Proveedores que suministran Artículos azules a la Fábrica F1.
$t_{(1} \mid \exists x\, \exists a(x.NP=t \wedge x.NF=F1 \wedge x.NA=a.NA \wedge a.COLOR=Azul)).$

3.10) Artículos suministrados a las Fábricas de Madrid.
$t_{(1} \mid \exists x\, \exists f(x.NA=t \wedge x.NF=f.NF \wedge f.CIUDADF=Madrid)).$

3.11) Proveedores que suministran algún Artículo azul a las Fábricas de Madrid o Lisboa.
$t_{(1} \mid \exists x\, \exists f\, \exists a(x.NP=t \wedge x.NA=a.NA \wedge x.NF=f.NF \wedge a.COLOR=Azul \wedge (f.CIUDADF=Madrid \vee f.CIUDADF=Lisboa)).$

3.12) Artículos suministrados por Proveedores en cuya ciudad hay alguna Fábrica.
$t_{(1} \mid \exists x\, \exists p\, \exists f(x.NA=t \wedge x.NP=p.NP \wedge f.CIUDADF=p.CIUDADP)).$

3.13) Artículos suministrados a las Fábricas de Madrid por Proveedores de Madrid.
$t_{(1} \mid \exists x\, \exists p\, \exists f(x.NA=t \wedge x.NP=p.NP \wedge x.NF=f.NF \wedge p.CIUDADP=Madrid \wedge f.CIUDADF=Madrid).$

3.14) Fábricas suministradas por al menos un Proveedor de distinta ciudad.
$t_{(1} \mid \exists x\, \exists p\, \exists f(x.NF=t \wedge x.NP=p.NP \wedge x.NF=f.NF \wedge p.CIUDADP \neg = f.CIUDADF).$

3.15) Fábricas que no son suministradas de Artículos azules por Proveedores de Madrid.

$t_{(1} \mid (\exists f(f.NF = t)) \wedge \neg(\exists x \exists p \exists a(x.NF = t \wedge x.NA = a.NA \wedge x.NP = p.NP \wedge a.COLOR = Azul \wedge p.CIUDADP = Madrid)).$

3.16) Proveedores que suministran al menos un Artículo suministrado por al menos otro Proveedor que suministra al menos un Artículo azul.

$t_{(1} \mid \exists x \exists y \exists z \exists a(x.NP = t \wedge x.NA = y.NA \wedge y.NP = z.NP \wedge z.NA = a.NA \wedge a.COLOR = Azul).$

3.17) Fábricas que usan al menos un Artículo suministrado por el Proveedor P1.

$t_{(1} \mid \exists x \exists y(x.NF = t \wedge x.NA = y.NA \wedge y.NP = P1).$

3.18) Parejas de ciudades tales que un Proveedor de la primera abastece a una Fábrica de la segunda.

$t_{(2} \mid \exists p \exists f \exists x(p.CIUDADP = t1 \wedge f.CIUDADF = t2 \wedge p.NP = x.NP \wedge x.NF = f.NF).$

3.19) Obtener las tripletas de valores de ⟨CIUDAD, NA, CIUDAD⟩ tales que un Proveedor de la primera ciudad abastece el artículo NA a una Fábrica de la segunda ciudad.

$t_{(3} \mid \exists p \exists f \exists x(p.CIUDADP = t1 \wedge f.CIUDADF = t3 \wedge p.NP = x.NP \wedge x.NF = f.NF \wedge x.NA = t2).$

3.20) Repetir la pregunta anterior, pero sin obtener las tripletas en que ambas ciudades son la misma.

$t_{(3} \mid \exists p \exists f \exists x(p.CIUDADP = t1 \wedge f.CIUDADF = t3 \wedge p.NP = x.NP \wedge x.NF = f.NF \wedge x.NA = t2 \wedge t1 \neg = t3).$

3.21) Proveedores que suministran un mismo Artículo, al menos, a todas las Fábricas.

$t_{(1} \mid (\exists p(p.NP = t)) \wedge (\exists a \exists f \exists x(x.NP = t \wedge x.NA = a.NA \wedge x.NF = f.NF)).$

3.22) Fábricas que tienen como único Proveedor a P1.

$t_{(1} \mid (\exists f(f.NF = t)) \wedge (\forall x(x.NF \neg = t \vee (x.NF = t \wedge x.NP = P1)).$

3.23) Artículos que son suministrados a todas las Fábricas de Madrid (excluyendo los que sólo se suministran a algunas).

$t_{(1} \mid (\exists a(a.NA = t)) \wedge (\forall f \exists x(f.CIUDADF \neg = Madrid \vee (f.CIUDADF = Madrid \wedge x.NA = t \wedge x.NF = f.NF)).$

3.24) Fábricas que usan, al menos, todos los Artículos suministrados por el Proveedor P1.

$t_{(1} \mid (\exists f(f.NF = t)) \wedge (\forall x \exists y(x.NP \neg = P1 \vee (x.NP = P1 \wedge y.NF = t \wedge y.NA = x.NA)).$

3.25) Fábricas que usan sólo Artículos que pueden ser suministrados por el Proveedor P1.

$t_{(1} \mid (\exists f(f.NF = t)) \wedge (\forall x(x.NF \neg = t \vee (x.NF = t \wedge \exists y(y.NA = x.NA \wedge y.NP = P1)))).$

3.26) Fábricas abastecidas por el Proveedor P1 con todos los Artículos que éste suministra.

$t_{(1} \mid \forall x \exists y(x.NP \neg = P1 \vee (x.NP = P1 \wedge y.NA = x.NA \wedge y.NP = P1 \wedge y.NF = t)).$

3.27) Fábricas que obtienen del Proveedor P1, total o parcialmente, todos los Artículos que usan.

$t_{(1} \mid (\exists f(f.NF = t)) \wedge (\forall x \exists y(x.NF \neg = t \vee (x.NF = t \wedge y.NP = P1 \wedge y.NA = x.NA \wedge y.NF = t))).$

F

3.28) Fábricas abastecidas por todos los Proveedores que suministran algún Artículo de color azul.

$t_{(1} \mid (\exists f(f.NF = t)) \wedge (\forall x((\neg \exists y \exists a(y.NP = x.NP \wedge y.NA = a.NA \wedge a.COLOR = Azul)) \vee ((\exists y \exists a(y.NP = x.NP \wedge y.NA = a.NA \wedge a.COLOR = Azul)) \wedge (\exists z(z.NP = x.NP \wedge z.NF = t)))))$.

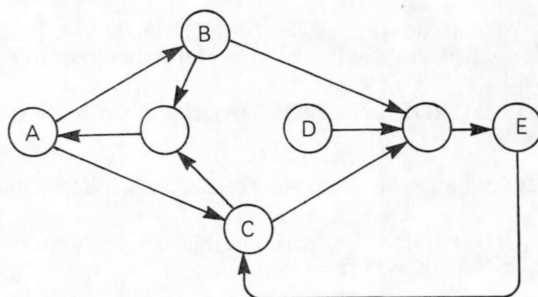
NORMALIZACION

1) *Sea el esquema R(A, B, C, D, E) con las dependencias:*

A→BC; BC→A; BCD→E; E→C.

Normalizar R en FNBC.

Diagrama de dependencias funcionales:



Claves: (AD), (BCD), (BDE).

Proyectando sobre los atributos de BC→A y E→C, en este orden, se produce la descomposición siguiente:

- R1 (A, B, C).
  A→B; A→C; BC→A.
  Claves: (A), (BC).
  FNBC.
- R2 (B, C, D, E).
  BCD→E; E→C.
  Claves: (BCD), (BDE).
- R21 (E, C).
  E→C.
  Clave: (E).
  FNBC.
- R22 (B, D, E).
  Clave: (BDE).
  FNBC.

El resultado de la descomposición es: R1 (A, B, C), R21 (E, C) y R22 (B, D, E). En el proceso se ha perdido la dependencia BCD→E.

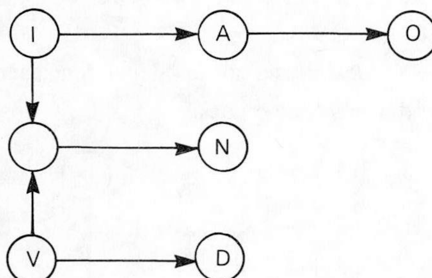
F

2) *Sean los atributos A (Agente), O (Oficina), I (Inversor), V (Clase de valor), N (Nominal) y D (Dividendo), y el esquema de relación R(A, O, I, V, N, D), con las siguientes dependencias:*

V→D; I→A; IV→N; A→O.

Normalizar R en FNBC.

Diagrama de dependencias funcionales:



Clave: (IV).

Proyectamos sobre los atributos de A→O:

- R1 (A, O).
  A→O.
  Clave: (A).
  FNBC.
- R2 (A, I, V, N, D).
  IV→N; I→A; V→D.
  Clave: (IV).

Proyectamos R2 sobre los atributos de V→D:

- R21 (V, D).
  V→D.
  Clave: (V).
  FNBC.
- R22 (A, I, V, N).
  IV→N; I→A.
  Clave: (IV).

Proyectamos R22 sobre los atributos de I→A:

- R221 (I, A).
  I→A.
  Clave: (I).
  FNBC.

- R222 (I, V, N).
  IV→N.
  Clave: (IV).
  FNBC.

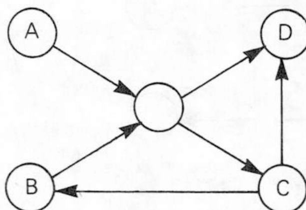
El resultado de la descomposición es: R1 (A, O), R21 (V, D), R221 (I, A) y R222 (I, V, N). Además se han preservado las dependencias.

3) *Sea el esquema R(A, B, C, D) con las dependencias:*

   AB→D; C→D; AB→C; C→B.

   a) Normalizar R en FN3. ¿Se preservan las dependencias?

   Diagrama de dependencias funcionales:



   Claves: (AB), (AC).

   Proyectando sobre los atributos de C→D:

   - R1 (C, D).
     C→D.
     Claves: (C).
     FNBC.
   - R2 (A, B, C).
     AB→C; C→B.
     Claves: (AB), (AC).
     FN3 (pero no FNBC).

   El resultado de la descomposición es: R1 (C, D) y R2 (A, B, C).

   Aunque la dependencia AB→D desaparece en el proceso, no se pierde, pues es inferible de las que se han preservado: de AB→C y C→D se deduce por transitividad que AB→D.

   b) Normalizar R en FNBC. ¿Se preservan las dependencias?

   Proyectando la R2 anterior sobre los atributos de C→B:

   - R21 (B, C).
     C→B.
     Claves: (C).
     FNBC.
   - R22 (A, C).
     Claves: (AC).
     FNBC.

El resultado de la descomposición es: R1 (C, D), R21 (B, C) y R22 (A, C). Se ha perdido la dependencia AB→C.

c) Aplicar a R el algoritmo que preserva dependencias. Comprobar que se obtienen relaciones FN3.

Como AB→D es redundante, sólo consideramos las dependencias: AB→C y C→BD.

Proyectando sobre ellas encontramos:

- R1 (A, B, C).
  AB→C; C→B.
  Claves: (AB), (AC).
  FN3 (no FNBC).
- R2 (B, C, D).
  C→D; C→B.
  Claves: (C).
  FNBC.

Proyectando sobre la clave (AB):

R3 (A, B).
Claves: (AB).
FNBC.
Redundante con R1.

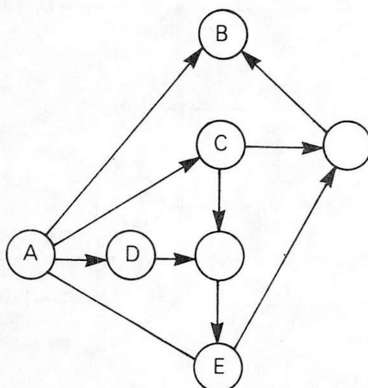
El resultado de la descomposición es: R1 (A, B, C) y R2 (B, C, D).

4) *Sea el esquema R (A, B, C, D, E) con las dependencias:*

A→BCDE; CD→E; CE→B.

Normalizar R en FNBC. ¿Se preservan las dependencias?

Diagramas de dependencias funcionales:



Claves: (A).

Proyectando sobre los atributos de CD→E y CD→B, en este orden, se produce la descomposición:

- R1 (C, D, E).
  CD→E.
  Claves: (CD).
  FNBC.
- R2 (A, B, C, D).
  A→BCD; CD→B.
  Claves: (A).
- R21 (B, C, D).
  CD→B.
  Claves: (CD).
  FNBC.
- R22 (A, C, D).
  A→CD.
  Claves: (A).
  FNBC.

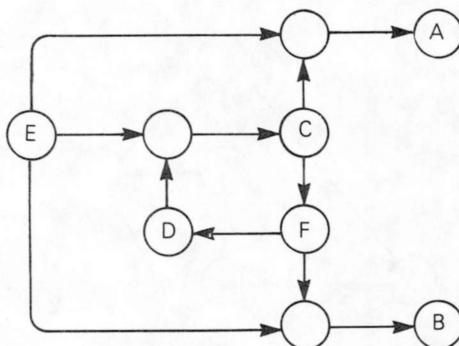
El resultado de la descomposición es: R1 (C, D, E), R21 (B, C, D) y R22 (A, C, D). En el proceso han desaparecido las dependencias: A→B, A→E y CE→B. Las dos primeras son inferibles de las que se han preservado, pero la tercera no. Por tanto, hay pérdida de dependencias.

5) *Sea el esquema R (A, B, C, D, E, F) con las dependencias:*

DE→C; C→F; F→D; CE→A; EF→B.

a) Normalizar R en FNBC. ¿Se preservan las dependencias?

Diagramas de dependencias funcionales:



Claves: (CE), (DE), (EF).

Proyectando sobre los atributos de C→F y C→D, en este orden, se produce la descomposición:

F

- R1 (C, F).
  C→F.
  Claves: (C).
  FNBC.
- R2 (A, B, C, D, E).
  C→D; CE→A; CE→B; DE→C.
  Claves: (CE), (DE).
- R21 (C, D).
  C→D.
  Claves: (C).
  FNBC.
- R22 (A, B, C, E).
  CE→A; CE→B.
  Claves: (CE).
  FNBC.

El resultado de la descomposición es: R1 (C, F), R21 (C, D) y R22 (A, B, C, E). En el proceso han desaparecido las dependencias: DE→C, F→D y EF→B. No se preservan las dependencias por tanto.

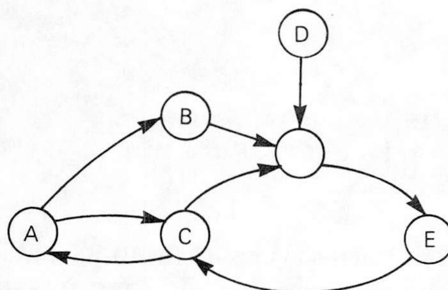
b) Normalizar R en FN3. ¿Se preservan las dependencias?
R ya está en FN3.

6) *Sea el esquema R (A, B, C, D, E) con las dependencias:*

C→A; A→BC; BCD→E; E→C.

a) Normalizar R en FNBC. ¿Se preservan las dependencias?

Diagrama de dependencias funcionales:



Claves: (AD), (CD), (DE).

Proyectando sobre los atributos de A→C, E→C y E→B, en este orden, se produce la descomposición:

- R1 (A, C).
  A→C; C→A.
  Claves: (A), (C).
  FNBC.

F

- R2 (B, C, D, E).
  $C \rightarrow B$; $E \rightarrow C$; $BCD \rightarrow E$.
  Claves: (CD), (DE).
- R21 (C, E).
  $E \rightarrow C$.
  Claves: (E).
  FNBC.
- R22 (B, D, E).
  $E \rightarrow B$.
  Claves: (DE).
- R221 (B, E).
  $E \rightarrow B$.
  Claves: (E).
  FNBC.
- R222 (D, E).
  Claves: (D, E).
  FNBC.

El resultado de la descomposición es: R1 (A, C), R21 (C, E), R221 (B, E) y R222 (D, E).

En el proceso han desaparecido las dependencias: $A \rightarrow B$ y $BCD \rightarrow E$. No se preservan las dependencias por tanto.

b) Normalizar R en FN3. ¿Se preservan las dependencias?

Proyectando sobre los atributos de $A \rightarrow B$ se produce la descomposición:

- R1 (A, B).
  $A \rightarrow B$.
  Claves: (A).
  FNBC.
- R2 (A, C, D, E).
  $A \rightarrow C$; $C \rightarrow A$; $E \rightarrow C$; $CD \rightarrow E$.
  Claves: (AD), (CD), (DE).
  FN3.

El resultado de la descomposición es: R1 (A, B) y R2 (A, C, D, E).

Se preservan las dependencias, pues aunque $BCD \rightarrow E$ ha desaparecido, es inferible de las que quedan. En efecto: $BCD \rightarrow CD \rightarrow E$, luego $BCD \rightarrow E$.

c) Aplicar a R el algoritmo que preserva dependencias. Comprobar que se obtienen relaciones FN3.

Empecemos substituyendo $BCD \rightarrow E$ por $CD \rightarrow E$, para tener un cubrimiento mínimo (eliminamos el atributo B que es redundante). Partimos por tanto de: $C \rightarrow A$; $A \rightarrow BC$; $CD \rightarrow E$; $E \rightarrow C$. Se produce la descomposición:

- R1 (A,C).
  $A \rightarrow C$; $C \rightarrow A$.

F

Claves: (A), (C).
FNBC.

- R2 (A, B, C).
  A→B; A→C; C→A.
  Claves: (A), (C).
  FNBC.
- R3 (C, D, E).
  CD→E; E→C.
  Claves: (CD), DE).
  FN3.
- R4 (C, E).
  E→C.
  Claves: (E).
  FNBC.
- R5 (A, D).
  Claves: (AD).
  FNBC.

Como R4 es redundante con R3, el resultado es: R1 (A, C), R2 (A, B, C), R3 (C, D, E) y R5 (A, D).

Podemos comprobar que esta descomposición es reversible por yunción. Para ello representemos como tablero la DY∗(AC, ABC, CDE, AD), y apliquémosle las dependencias dadas. Partimos del tablero:

A	B	C	D	E
a	—	c	—	—
a	b	c	—	—
—	—	c	d	e
a	—	—	d	—
a	b	c	d	e

Apliquemos C→A:

A	B	C	D	E
a	—	c	—	—
a	b	c	—	—
a	—	c	d	e
a	—	—	d	—

Apliquemos A→BC:

A	B	C	D	E
a	b	c	—	—
a	b	c	—	—
a	b	c	d	e
a	b	c	d	—

Vemos que aparece la fila ⟨abcde⟩. Por tanto, es reversible.

F

7) *Añadir a la relación siguiente las filas necesarias para que se cumplen en ella las DPs: A→»BC; CD→»BE.*

A	B	C	D	E
a	b	c	d	e
a	1	c	2	e
3	b	c	d	4

Hay que añadir las filas: ⟨abc2e⟩, ⟨a1cde⟩, ⟨abcd4⟩, ⟨a1cd4⟩, ⟨3bcde⟩, ⟨31cde⟩ y ⟨31cd4⟩.

8) *Sea el esquema R (A, B, C, D, E) con las dependencias:*

A→»BC; DE→»C.

Demostrar que AD→»BE.

De A→»BC, por complementación, se infiere: A→»DE.

De A→»DE y DE→»C, por transitividad: A→»(C – DE), es decir, A→»C.

De AD→»A (reflexividad), y A→»C, por transitividad: AD→»C.

De AD→»C, por complementación: AD→»BE, c.q.d.

También podemos utilizar el algoritmo de batida y la representación de AD→»BE mediante tableros. En efecto, esta dependencia es representable como la DY * (ADBE, ADC). Representémosla con un tablero:

A	B	C	D	E
a	b	1	d	e
a	2	c	d	3
a	b	c	d	e

Apliquémosle A→»BC:

A	B	C	D	E
a	b	1	d	e
a	2	c	d	3
a	b	1	d	3
a	2	c	d	e

Apliquémosle DE→»C:

A	B	C	D	E
a	b	1	d	e
a	2	c	d	3
a	b	1	d	3
a	2	c	d	e
a	2	1	d	e
a	b	c	d	e

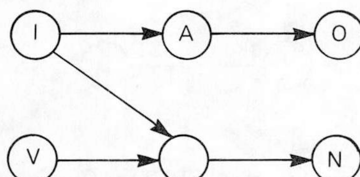
....

Vemos que aparece la fila ⟨abcde⟩, c.q.d.

9) *En el esquema del ejercicio 2 anterior queremos guardar la historia de los dividendos de cada tipo de valor. Entonces, en vez de cumplirse V→D, tendremos V↠D.*

Normalizar R en FN4. ¿Se preservan las dependencias?

Diagrama de dependencias funcionales:



Claves: (IV).

Proyectando sobre los atributos de V↠D, A→O e I→A, en este orden, se produce la descomposición:

- R1 (V, D).
  Claves: (VD).
  FNBC, y también FN4 al no haber DPs.
- R2 (A, O, I, V, N).
  I→A; A→O; IV→N.
  Claves: (IV).
- R21 (A, O).
  A→O.
  Claves: (A).
  FNBC (y FN4).
- R22 (A, I, V, N).
  I→A; IV→N.
  Claves: (IV).
- R221 (I, A).
  I→A.
  Claves: (I).
  FNBC (y FN4).
- R222 (I, V, N).
  IV→N.
  Claves: (IV).
  FNBC (y FN4).

El resultado de la descomposición es: R1 (V, D), R21 (A, O), R221 (I, A) y R222 (I, V, N).

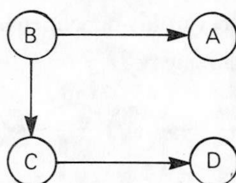
Se han preservado las dependencias.

10) *Sea el esquema R (A, B, C, D, E, F) con las dependencias:*

A↠BCD; B→AC; C→D.

Normalizar R en FN4. ¿Se preservan las dependencias?

Diagrama de dependencias funcionales:



Claves: (B).

Proyectando sobre los atributos de A→»BCD y C→D, en este orden, se produce la descomposición siguiente:

- R1 (A, B, C, D).
  B→A; B→C; C→D.
  Claves: (B).
- R2 (A, E, F).
  Claves: (AEF).
  FNBC y FN4.
- R11 (C, D).
  C→D.
  Claves: (C).
  FNBC y FN4.
- R12 (A, B, C).
  B→A; B→C.
  Claves: (B).
  FNBC y FN4.

Se ha llegado así al esquema R11 (C, D), R12 (A, B, C) y R2 (A, E, F).

Se preservan las dependencias.

11) *Sea el esquema R (A, B, C, D, E) con las dependencias:*

A→»BC; D→C.

a) Demostrar que A→C.

De A→»BC y D→C, y teniendo en cuenta que BC y D no tienen atributos comunes, y que C está contenido en BC, se deduce que A→C.

También puede deducirse aplicando el algoritmo de batida.

Tablero inicial (A→C):

A	B	C	D	E
a	1	b	3	5
a	2	c	4	6

b=c

Aplicamos A→»BC:

A	B	C	D	E
a	1	b	3	5
a	2	c	4	6
a	2	c	3	5
a	1	b	4	6

Aplicando D→C se substituye c por b:

A	B	C	D	E
a	1	b	3	5
a	2	b	4	6
a	2	b	3	5
a	1	b	4	6

Vemos que siempre que se repite A se repite también C, luego A→C, c. q. d.

b) Normalizar R en FN4. ¿Se pierden dependencias?

Claves: (ABDE).

Proyectando sobre los atributos de A→»BC y A→C, en este orden, se produce la descomposición siguiente:

- R1 (A, B, C).
  A→C.
  Claves: (AB).
- R2 (A, D, E).
  Claves: (ADE).
  FNBC y FN4.
- R11 (A, C).
  A→C.
  Claves: (A).
  FNBC y FN4.
- R12 (A, B).
  Claves: (AB).
  FNBC y FN4.

Se ha llegado así al esquema R11 (A, C), R12 (A, B) y R2 (A, D, E).

Se ha perdido la dependencia D→C.

12) *Sea la siguiente extensión de R (A, B, C):*

A	B	C
a	1	A
a	2	B
b	3	C
c	3	D
d	4	E
e	5	E

a) ¿Satisface esta extensión de R la DY: ∗(AB, AC, BC)?

Si se proyecta sobre AB, AC y BC, y se ayuntan estas proyecciones, se recompone la extensión dada. Por tanto, ésta satisface la DY.

b) Añadimos a esta extensión la fila ⟨f1B⟩. Añadir las filas que sean necesarias para que se siga cumpliendo la DY anterior.

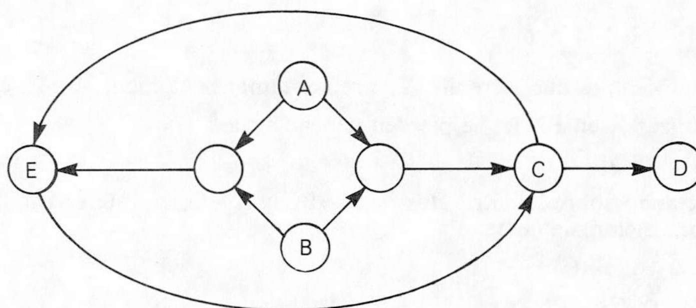
Hay que añadir la fila ⟨a1B⟩.

13) *Sea el esquema R (A, B, C, D, E) con las dependencias:*

AB→C; C→E; E→C; C→D; AB→E.

a) ¿Está R en FNBC?

Diagrama de dependencias funcionales:



Claves: (AB). No está en FNBC.

b) Descomponemos R por proyecciones en R1 (A, B, C), R2 (A, D, E) y R3 (C, E). ¿Es esta descomposición reversible por yunción?

Para que lo sea, la DY ∗(ABC, ADE, CE) tiene que ser inferible de las dependencias funcionales dadas. Para comprobar si es así puede utilizarse el algoritmo de batida.

Tablero inicial:

A	B	C	D	E
a	b	c	—	—
a	—	—	d	e
—	—	c	—	e
a	b	c	d	e

Apliquémosle la dependencia C→E:

A	B	C	D	E
a	b	c	—	e
a	—	—	d	e
—	—	c	—	e
a	b	c	d	e

Apliquémosle la dependencia E→C:

A	B	C	D	E
a	b	c	—	e
a	—	c	d	e
—	—	c	—	e
a	b	c	d	e

Apliquémosle la dependencia C→D:

A	B	C	D	E
a	b	c	d	e
a	—	c	d	e
—	—	c	d	e
a	b	c	d	e

Aparece en el tablero la fila ⟨abcde⟩. Por tanto, la DY es inferible de las dependencias funcionales dadas.

c) ¿Preserva las dependencias?

Dependencias preservadas en el esquema final:

R1: AB→C.
R2: E→D.
R3: C→E; E→C.

Dependencias que han desaparecido: AB→E y C→D. Pero son inferibles de las preservadas, por lo que no se pierden dependencias.

d) Normalizar R1, R2 y R3 en FNBC.

R1 está ya en FNBC, pues su clave es (AB).

Lo mismo ocurre con R3, cuyas claves son (C) y (E).

R2 la descomponemos proyectando sobre los atributos de E→D:

- R21 (D, E).
  E→D.
  Claves: (E).
  FNBC.
- R22 (A, E).
  Claves: (AE).
  FNBC.

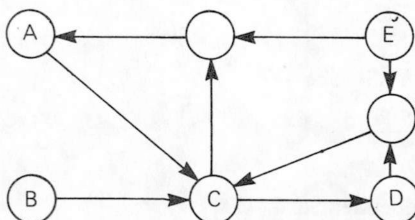
Se ha llegado así al esquema R1 (A, B, C), R21 (D, E), R22 (A, E) y R3 (C, E). Se han preservado las dependencias.

14) *Sea el esquema R (A, B, C, D, E) con las dependencias:*

A→C; B→C; C→D; DE→C; CE→A; ∗(AD, AB, BE, CDE, AE).

a) Hallar las claves.

Diagrama de dependencias funcionales:



Claves: (BE).

b) ¿Es la DY inferible de las DFs?

Para comprobar si es así utilizaremos el algoritmo de batida.

Tablero inicial:

A	B	C	D	E
a	—	1	d	—
a	b	—	—	—
—	b	—	—	e
—	—	c	d	e
a	—	—	—	e
a	b	c	d	e

Apliquémosle la dependencia A→C:

A	B	C	D	E
a	—	1	d	—
a	b	1	—	—
—	b	—	—	e
—	—	c	d	e
a	—	1	—	e
a	b	c	d	e

Apliquémosle la dependencia B→C:

A	B	C	D	E
a	—	1	d	—
a	b	1	—	—
—	b	1	—	e
—	—	c	d	e
a	—	1	—	e
a	b	c	d	e

Apliquémosle la dependencia C→D:

A	B	C	D	E
a	—	1	d	—
a	b	1	d	—
—	b	1	d	e
—	—	c	d	e
a	—	1	d	e
a	b	c	d	e

Apliquémosle la dependencia DE→C:

A	B	C	D	E
a	—	c	d	—
a	b	c	d	—
—	b	c	d	e
—	—	c	d	e
a	—	c	d	e
a	b	c	d	e

Apliquémosle la dependencia CE→A:

A	B	C	D	E
a	—	c	d	—
a	b	c	d	—
a	b	c	d	e
a	—	c	d	e
a	—	c	d	e
a	b	c	d	e

Aparece en el tablero la fila ⟨abcde⟩. Por tanto, la DY es inferible de las dependencias dadas.

c) Las dependencias definidas por las claves son: BE→ABCDE.

Si las aplicamos al tablero inicial éste no cambia. Por tanto, la DY no es inferible de las claves.

A la misma conclusión se llega directamente viendo que ninguna intersección entre los componentes de la DY: AD, AB, BE, CDE y AE, contiene a la clave.

d) ¿Está R en FN5?

No, puesto que la DY no es inferible de las claves. O, desde otro punto de vista, como no está en FNBC tampoco puede estar en FN5.

15) *Sea el esquema R (A, B, C, D, E) con las claves:*

K1: (BCE); K2: (ABE).

Decir si son inferibles de las claves las DYs siguientes:

a) *(ABCE, BCDE).

F

Las dependencias definidas por las claves son:

BCE→ABCDE y ABE→ABCDE. Apliquemos una batida.

Tablero inicial:

A	B	C	D	E
a	b	c	—	e
—	b	c	d	e
a	b	c	d	e

Apliquémosle la dependencia BCE→ABCDE:

A	B	C	D	E
a	b	c	d	e
a	b	c	d	e
a	b	c	d	e

Aparece en el tablero las fila ⟨abcde⟩. Por tanto, la DY es inferible de las claves. Podemos deducir esto mismo mediante el algoritmo de Fagin. Como ambos componentes de la DY incluyen a la calve K1 los reagrupamos obteniendo la DY * (ABCDE). Esta DY incluye todos los atributos de R. Por tanto la DY dada es inferible de las claves.

b) *(ABCE, BDE, ACD).

Al aplicar las dependencias al tablero éste no cambia. Por tanto la DY no es inferible de las claves.

A la misma conclusión se llega directamente viendo que ninguna intersección entre los componentes de la DY: ABCE, BDE y ACD, contiene a una clave.

c) * (ABCE, BCE, ABDE).

Apliquemos una batida. Tablero inicial:

A	B	C	D	E
a	b	c	—	e
—	b	c	—	e
a	b	—	d	e
a	b	c	d	e

Apliquémosle la dependencia BCE→ABCDE:

A	B	C	D	E
a	b	c	—	e
a	b	c	—	e
a	b	—	d	e
a	b	c	d	e

Apliquémosle la dependencia ABE→ABCDE:

A	B	C	D	E
a	b	c	d	e
a	b	c	d	e
a	b	c	d	e
a	b	c	d	e

Aparece en el tablero la fila ⟨abcde⟩. Por tanto, la DY es inferible de las claves.

Apliquemos el algoritmo de Fagin. Los dos primeros componentes incluyen a la clave K1. Los reagrupamos: *(ABCE, ABDE). Ahora incluyen a K2. Los reagrupamos: *(ABCDE). Esta DY incluye a todos los atributos de R. Por tanto la DY dada es inferible de las claves.

16) *En el esquema del ejercicio 2 anterior descomponemos R por proyecciones en R*1 *(V, D), R*2 *(I, A), R*3 *(I, V, D) y R*4 *(A, O). ¿Es esta descomposición reversible por yunción?*

No puede serlo porque no incluye todos los atributos (falta N).

17) *En el esquema del ejercicio 2 anterior descomponemos R por proyecciones en R*1 *(I, V, N), R*2 *(I, A), R*3 *(V, D) y R*4 *(I, V, O). ¿Es esta descomposición reversible por yunción? ¿Preserva las dependencias?*

Será reversible si la DY * (IVN, IA, VD, IVO) es inferible de las dependencias dadas.

Apliquemos una batida. Tablero inicial:

A	O	I	V	N	D
—	—	i	v	n	—
a	—	i	—	—	—
—	—	—	v	—	d
—	o	i	v	—	—
a	o	i	v	n	d

Apliquémosle la dependencia V→D:

A	O	I	V	N	D
—	—	i	v	n	d
a	—	i	—	—	—
—	—	—	v	—	d
—	o	i	v	—	d
a	o	i	v	n	d

Apliquémosle la dependencia I→A:

F

A	O	I	V	N	D
a	—	i	v	n	d
a	—	i	—	—	—
—	—	—	v	—	d
a	o	i	v	—	d
a	o	i	v	n	d

Apliquémosle la dependencia IV→N:

A	O	I	V	N	D
a	—	i	v	n	d
a	—	i	—	—	—
—	—	—	v	—	d
a	o	i	v	n	d
a	o	i	v	n	d

Aparece en el tablero la fila ⟨aoivnd⟩. Por tanto, la DY es inferible de las claves.

Dependencias preservadas:

- R1: IV→N.
- R2: I→A.
- R3: V→D.
- R4: I→O.

Ha desaparecido A→O. Por tanto, no se preservan las dependencias.

18) *Sea el esquema R (A, B, C, D, E). Los atributos (ABC) y (ABD) toman valores que no se repiten (es decir, son claves o superclaves). Además, se verifica AB→→C. Hallar una clave de R.*

Como AB→→C y ABD→C, y además C y ABD no tienen atributos comunes, se infiere que AB→C.

Por otra parte, de AB→→C se infiere, por complementación, que AB→→DE. De aquí y de ABC→D, y teniendo en cuenta además que D está contenido en DE, y que ABC y DE no tienen atributos comunes, se infiere que AB→D.

Análogamente, de AB→→DE y ABC→E, se infiere AB→E.

Por tanto, AB→ABCDE, luego (AB) es una clave de R.

Podemos comprobarlo mediante el algoritmo de batida.

Dependencias de partida:

ABC→DE; ABD→CE; AB→→C.

Dependencia que hay que estudiar: AB→CDE.

Tablero inicial:

A	B	C	D	E
a	b	c1	d1	e1
a	b	c2	d2	e2
		c1 = c2	d1 = d2	e1 = e2

F

Apliquemos la dependencia AB→→C:

A	B	C	D	E
a	b	c1	d1	e1
a	b	c2	d2	e2
a	b	c2	d1	e1
a	b	c1	d2	e2

Apliquemos la dependencia ABC→DE:

A	B	C	D	E
a	b	c1	d1	e1
a	b	c2	d1	e1
a	b	c2	d1	e1
a	b	c1	d1	e1

Apliquemos la dependencia ABD→CE:

A	B	C	D	E
a	b	c1	d1	e1
a	b	c1	d1	e1
a	b	c1	d1	e1
a	b	c1	d1	e1

Todas las veces que se repite AB, se repiten CDE. Por tanto, se verifica AB→CDE.

19) *Sea el esquema de relación MATRIM (H, M, S, FB, FD), con el significado siguiente:*

En MATRIM hay una fila por cada matrimonio celebrado en el país desde 1970.

Atributo H: Nombre del marido. Se supone que no hay personas diferentes con igual nombre.

Atributo M: Nombre de la mujer.
Atributo S: Situación del matrimonio. Puede ser una de las siguientes:

- C: Casados todavía.
- D: Ya divorciados.
- S: Separados.
- F: Alguno de los cónyuges, o ambos, han fallecido.

Atributo FB: Fecha de la boda.

Atributo FD: Fecha de disolución del matrimonio, bien por separación o divorcio, bien por fallecimiento de uno o ambos cónyuges.

Enunciar las condiciones de integridad que lógicamente deberán cumplir todas las extensiones válidas de MATRIM.

Deberán cumplirse las condiciones siguientes:

a) Los valores de FB y FD deben ser fechas válidas (por ejemplo, el número de mes debe estar entre 1 y 12, si el mes es 1 el número de días debe estar entre 1 y 31, etcétera.).

F

b) El atributo S sólo puede tomar uno de los valores C, D, S ó F.

c) Los atributos FB y FD deben ser fechas posteriores a 31 de diciembre de 1969.

d) En todas las filas en que el atributo S tiene el valor C, FD debe ser nulo. En las demás debe ser no nulo.

e) En todas las filas en que FD no es nulo, debe ser posterior a FB.

f) Si en el país son ilegales la poligamia y la poliandria, en las filas que tienen en S el valor C no pueden repetirse los valores de H o M. Dicho de otra forma, si seleccionamos todas las filas en que S tiene el valor C, los atributos H y M son claves.

g) Suponiendo que un cambio de sexo oblique legalmente a cambiar también el nombre, ningún valor de H puede repetirse en M, y recíprocamente.

h) Sea una fila cualquiera que tiene en el atributo S el valor F, y sean h, m y f los valores de sus columnas H, M y FB, respectivamente. Deberá cumplirse al menos una de las dos condiciones siguientes:

- Para toda otra fila en que H tenga el valor h, el valor de FD debe ser anterior a f.
- Para toda otra fila en que M tenga el valor m, el valor de FD debe ser anterior a f.

i) Supongamos que extraemos todas las filas en que se repite un valor cualquiera de H y las clasificamos por FB creciente. Deberán cumplir las condiciones siguientes:

- No puede haber más de una fila que tenga en S el valor C. Si la hay, debe ser la última.
- Suponiendo que legamente no puede volver a casarse una persona sin obtener previamente el divorcio o enviudar, no podrá haber más de una fila que tenga en el atributo S el valor S, y si la hay, debe ser la última.
- Toda fila deberá tener un valor en FB posterior al de FD en la fila precedente.

j) La condición anterior es también aplicable si consideramos M en vez de H.

k) Suponiendo que para la concesión del divorcio sea necesario un período previo de separación de un mínimo de dos años, deberá cumplirse que en todas las filas en que el atributo S contenga el valor D, el valor de FD deberá ser posterior al de FB en dos años o más.

l) Siempre que se actualicen valores en una fila deberán cumplirse las condiciones siguientes:

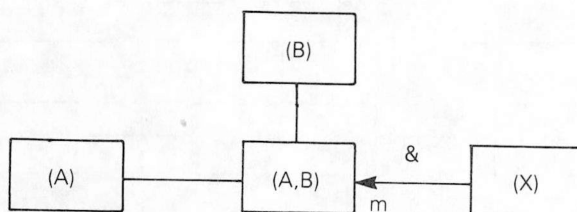
- Si el atributo S tiene el valor C, éste puede cambiar a S o F. En este caso también cambiará FD.
- Si el atributo S tiene el valor S, éste puede cambiar a C, D ó F. En este caso también cambiará FD, y si además S tenía S y cambia a D, el nuevo valor de FD deberá ser posterior al FD antiguo en dos o más años. Si S cambia del valor S al C, el nuevo valor de FD será nulo.
- Si el atributo S tiene el valor F, ningún campo de la fila puede cambiar.
- Si el atributo S tiene el valor D, ningún campo de la fila puede cambiar.
- Siempre que cambie FD, el nuevo valor deberá ser posterior al que tenía, si ambos valores, el viejo y el nuevo, son no nulos.

F

DIAGRAMAS PARA DISEÑO

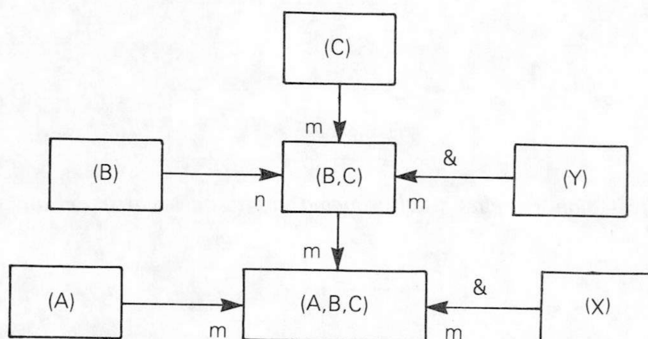
En los ejercicios siguientes, para simplificación de los dibujos, dejaremos a veces las asociaciones explícitas sin nombre. En este caso indicaremos que son explícitas adjuntándoles el signo &. También, por la misma razón, se omitirán a veces los nombres de los conjuntos de valores representados en los rectángulos.

1) *Obtener las relaciones y claves del esquema de diseño a partir del diagrama siguiente:*

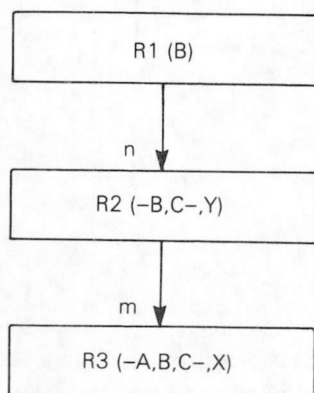


Aplicando las reglas de transformación se llega a la relación R (–A–, –B–, X).

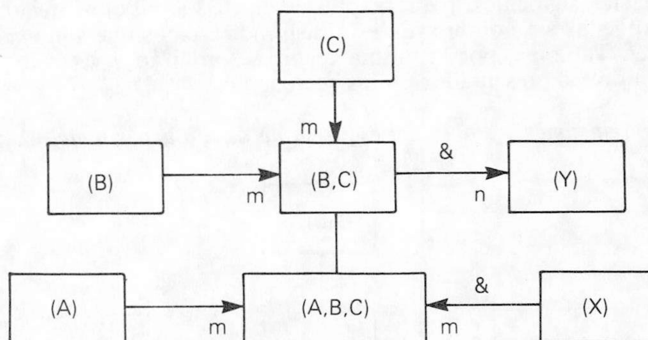
2) *Transformar el diagrama siguiente:*



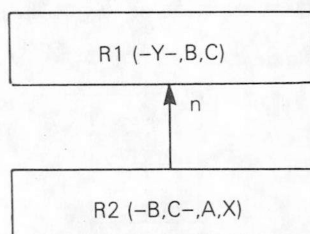
Se transforma en:



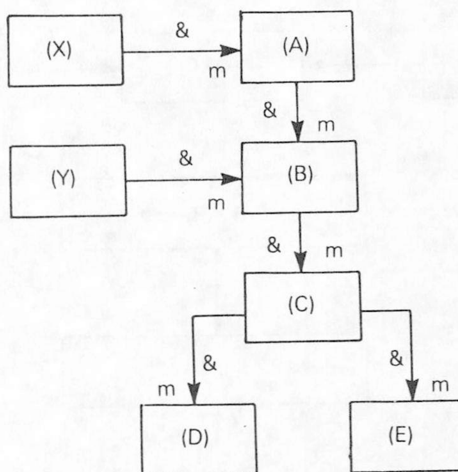
3) *Transformar el diagrama siguiente:*



Se transforma en:



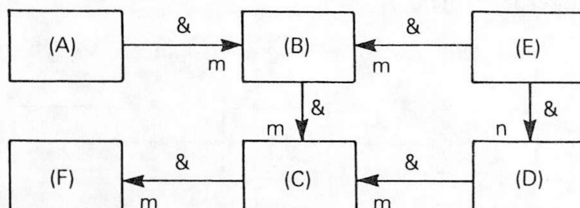
4) *Obtener las relaciones y claves del esquema de diseño a partir del diagrama siguiente:*



Aplicando las reglas de transformación, se llega a las relaciones siguientes:
R1 (–A–, X); R2 (–B–, A, Y); R3 (–C–, B); R4 (–D–, C); R5 (–E–, C).

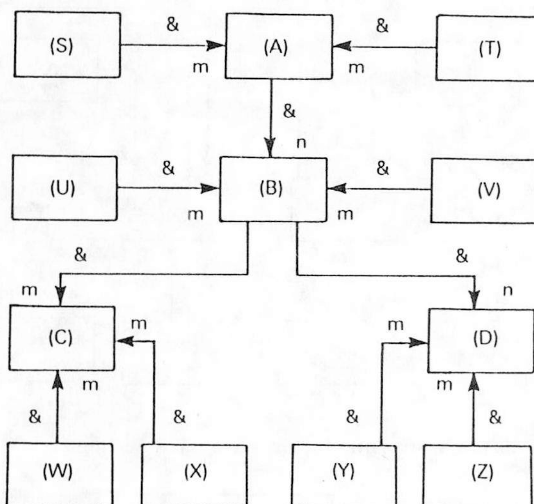
F

5) *Obtener las relaciones y claves del esquema de diseño a partir del diagrama siguiente:*

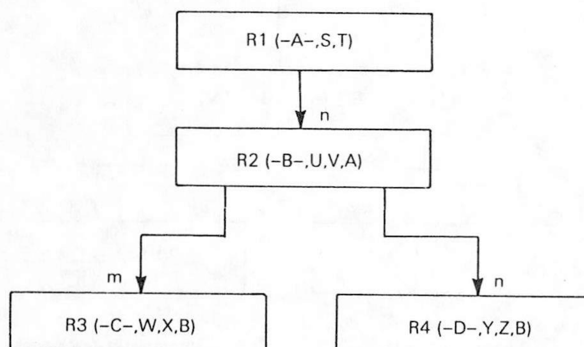


Aplicando las reglas de transformación, se llega a las relaciones siguientes:
R1 (–B–, A, E); R2 (–C–, B, D); R3 (–D–, E); R4 (–F–, C).

6) *Obtener las entidades definidas por el diagrama siguiente y decir de qué tipo son:*



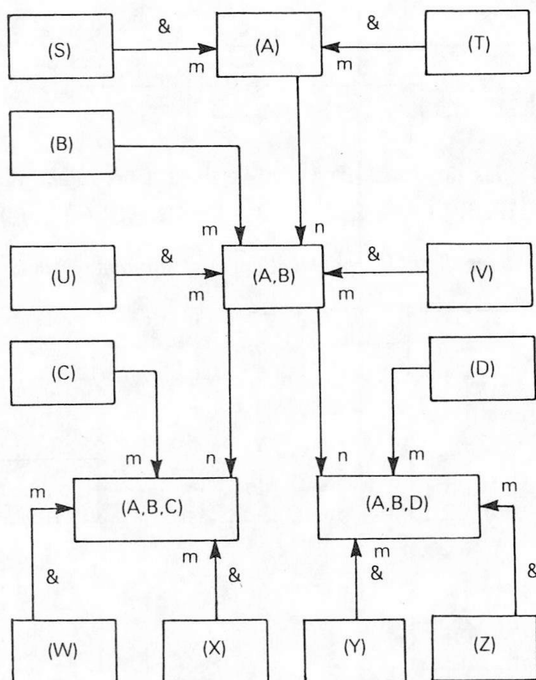
Propagando las clases y suprimiendo rectángulos se llega a:



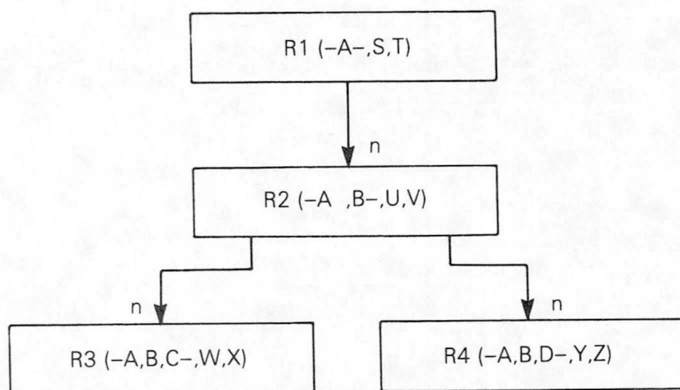
Entidades: R1 (Independiente); R2 (Asociativa); R3 y R4 (Dependientes).

R2 y R3 son interdependientes.

7) *Obtener las entidades definidas por el diagrama siguiente y decir en qué tipo son:*

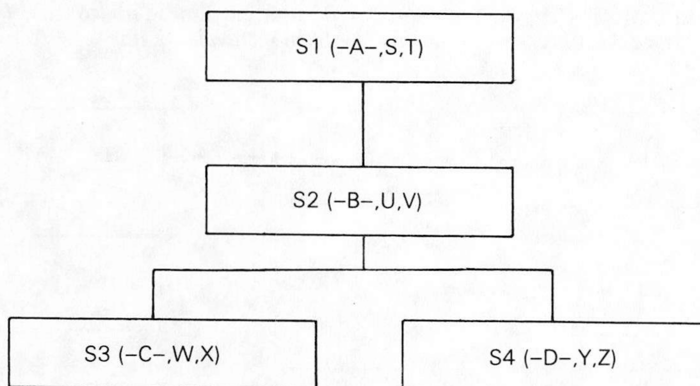


Propagando las claves y suprimiendo rectángulos se llega a:

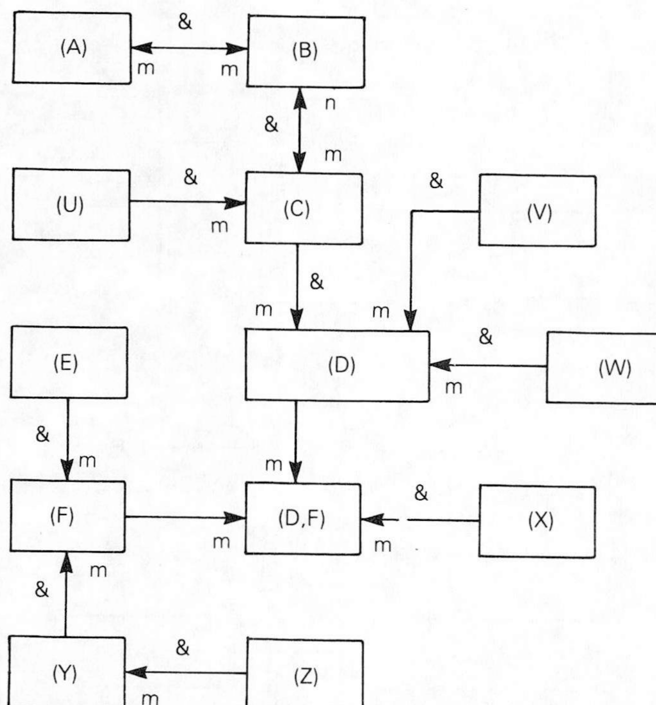


Entidades: R1 (Independiente); R2, R3 y R4 (Dependientes).

Expresión de este modelo en estructuras IMS:



8) *Obtener las relaciones y claves del esquema de diseño a partir del diagrama siguiente:*

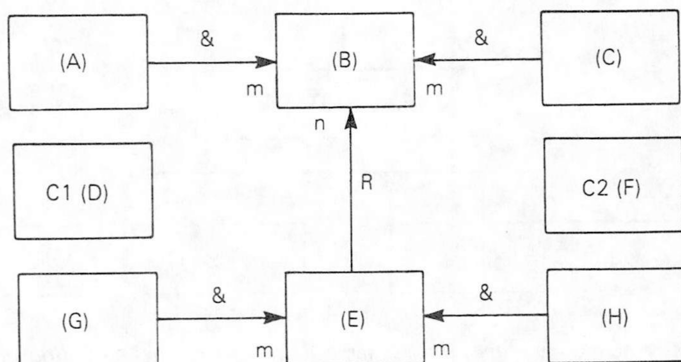


Transformando las asociaciones explícitas en implícitas y aplicando las reglas de transformación, se llega a las relaciones siguientes:

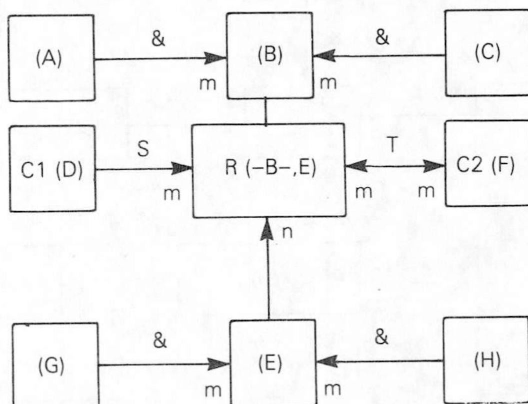
R1 (A, B); R2 (B, C); R3 (–C–, U); R4 (–Y–, Z); R5 (–F–, Y, E); R6 (–D, F–, X); R7 (–D–, C, V, W).

F

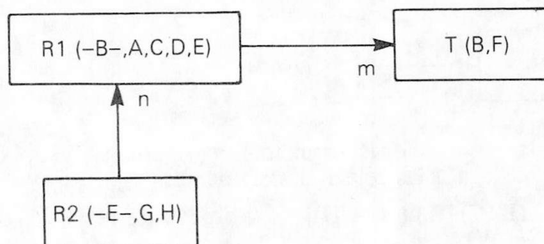
9) Partimos del diagrama siguiente. Además de las asociaciones representadas en él hay otras dos: S y T. La primera entre los elementos de C1 y R, con cardinalidad (1 : m). La segunda entre los elementos de C2 y R, con cardinalidad (m : m). *Representar S y T en el diagrama, obtener el esquema de diseño y decir de qué tipo son las Entidades obtenidas:*



Representamos R con un rectángulo:



El diagrama se transforma en el siguiente:



El diagrama relacional de diseño será el siguiente:

Relaciones: R1(–B–, A, C, D, E); R2(–E–, G, H); T (B, F).

Otras condiciones: **R1**[E]⊆**R2**[E]; **R1**[B]=**T**[B].

La Entidad R2 es fuerte, la T es débil y la R1 es de asociación. Además R1 y T son interdependientes.

10) Sean dos relaciones, A(X, Y, Z) y R(X, Y, W). La asociación implícita entre A y B según X es de cardinalidad (m : 1), según Y es de cardinalidad (m : c), y según (X, Y) es de cardinalidad (m : c). *Comprobar que estas cardinalidades no son contradictorias y poner un ejemplo de extensiones de ambas relaciones que las cumplan.*

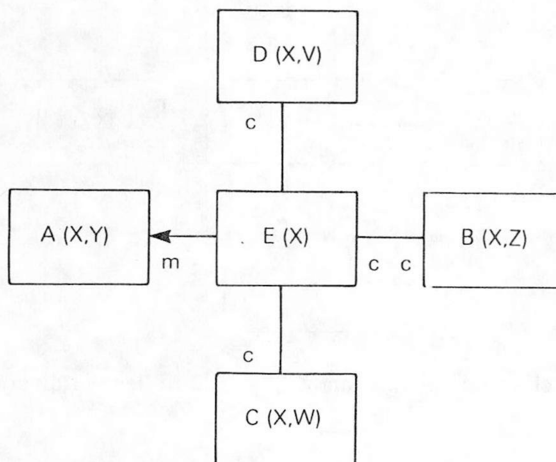
Según la tabla de cardinalidades de las asociaciones totales, combinando 1 con c se obtiene que la cardinalidad de A a B según (X, Y) puede valer c ó 0. Para la combinación m con m,la tabla da como posible cualquier valor (n, m, c, 1 ó 0). Por tanto, las cardinalidades dadas son posibles.

Ejemplo de valores:

A (X	Y	Z)
1	a	10
1	a	11
2	b	12
2	c	13

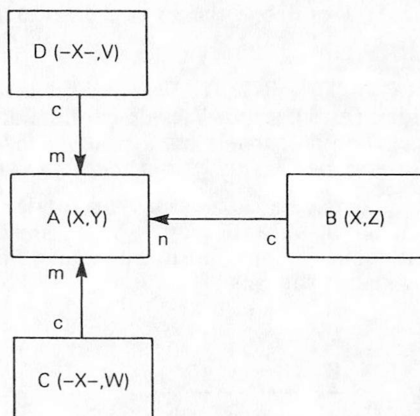
B (X	Y	W)
1	a	20
2	b	21

11) *Transformar el diagrama siguiente:*

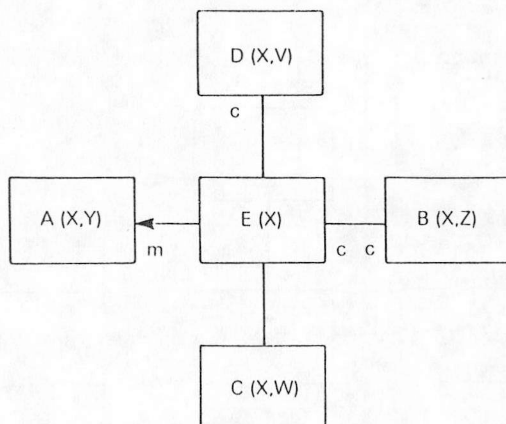


El rectángulo E participa con todos sus atributos en todas sus asociaciones, y en una de ellas con cardinalidad (m : 1). Por tanto, es suprimible. Para no perder información, hallamos las cardinaliades compuestas, mediante las tablas correspondientes. Así,la cardinalidad de la asociación implícita entre A y B, la hallamos componiendo las de (A-E) y (B-E). El resultado de aplicar las tablas es que de A a B, la cardinalidad puede valer c, 1 ó 0, y de B a A, puede valer n o c. Nos quedamos con los valores más generales, con lo que la cardinalidad compuesta entre A y B será (n : c).

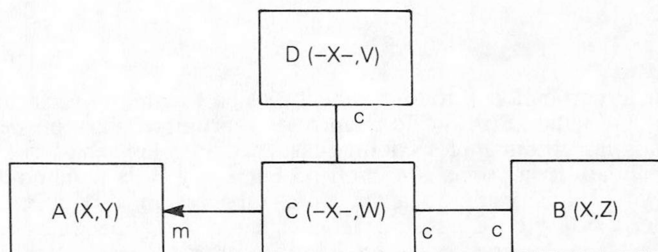
Análogamente, las cardinalidades compuestas entre A y C, y A y D, son (m:c). También podríamos hallar las cardinalidades entre C y D, C y B, y B y D, pero no es necesario porque son consecuencia de las anteriores. En conclusión, el diagrama queda transformado en el siguiente:



12) *Transformar el diagrama siguiente:*

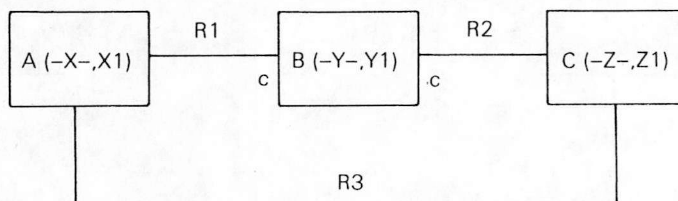


El atributo X es clave en C. Agregando C y E se obtiene el diagrama siguiente:



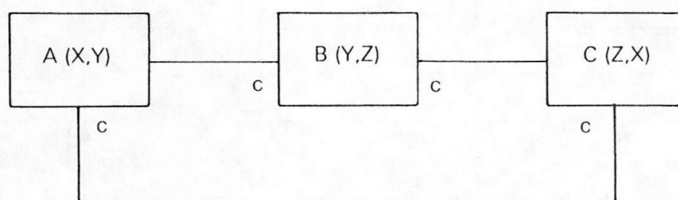
F

13) *Dado el siguiente diagrama, decir si la asociación explícita R3 es el producto de las asociaciones R1 y R2.*



Si R3 fuera el producto de R1 y R2, su cardinalidad vendría condicionada por las de éstas. Aplicando la tabla de producto de cardinalidades, tenemos que el camino A-R1-B-R2-C, combinando con 1, da como valor posible sólo c. Por tanto, R3 no puede ser el producto de R1 y R2.

14) *Decir si la asociación implícita entre A y C es redundante en el diagrama siguiente:*



Las cardinalidades de las asociaciones (A-B) y (B-C) no se pueden componer, pues intervienen en ellas atributos diferentes (Y y Z). La información aportada por el diagrama no permite afirmar si es redundante. En general no lo será. Ejemplo de valores:

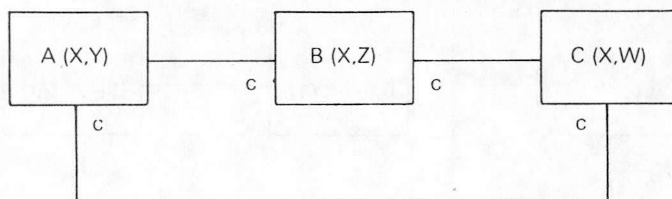
X	Y		Y	Z		Z	X
1	a		b	A		A	3
1	b		c	B		B	3
2	a		d	C		C	3
2	c					D	3
4	d					D	4

Si fuera redundante significaría que al ir de A a B y de B a C, se asociarían las mismas tuplas que yendo directamente de A a C. Por tanto sería necesario (no suficiente) que yendo de A a B y de B a C se obtuvieran iguales valores de X. Ejemplo:

X	Y		Y	Z		Z	X
1	b		b	A		A	1
2	c		c	B		B	2
4	d		d	C		C	4
5	a					D	6

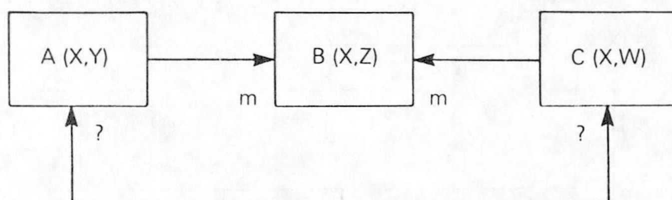
F

15) *Decir si la asociación implícita entre A y C es redundante en el diagrama siguiente:*



La asociación (A-C) puede obtenerse componiendo las asociaciones (A-B) y (B-C), pues tienen el mismo atributo X. Por otra parte, su cardinalidad, según las tablas, puede valer, en el sentido de A a C, n, m, c ó 1, y de C a A lo mismo. La cardinalidad dada es (c:c), que no contradice a estos valores. Sin embargo, si eliminamos del diagrama la asociación (A-C), perderíamos información sobre su cardinalidad. Por tanto, no es redundante incluirla en el dibujo.

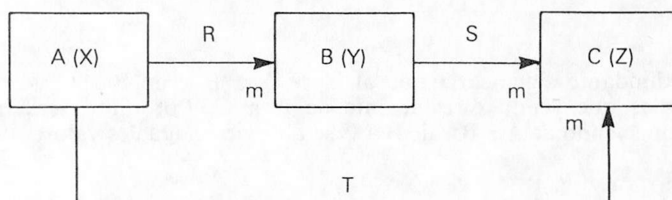
16) *En el diagrama siguiente, hallar las cardinalidades posibles de la asociación implícita entre A y C y decir si es redundante:*



Aplicando la tabla de composición de cardinalidades se obtiene que la cardinalidad entre A y C es (1:1). Por tanto, la cardinalidad está completamente definida, por lo que es redundante incluir la asociación entre A y C en el dibujo.

También se puede obtener la cardinalidad observando que X debe ser clave en A y en C (por la regla de Definición de Claves). Esto implica que las cardinalidades buscadas no pueden ser múltiples, es decir, han de ser 1 o c. Pero c no pueden ser, pues todos los valores de X en A existen en C, y viceversa. Luego han de ser 1.

17) *Sea el diagrama siguiente:*



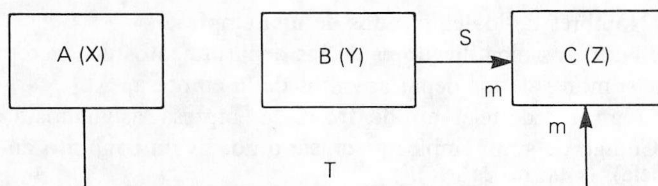
Supongamos que el bucle (R-S-T) se debe a que la asociación T es redundante con R y S. ¿Puede afirmarse también que R es redundante con S y T, y que S es redundante con R y T?

¿Cuál de las tres asociaciones, R, S o T, es preferible eliminar en el diagrama?

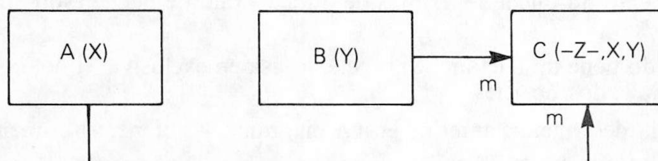
Puesto que T es redundante con R y S, si entramos por A, y tomamos un X, y recorremos el camino A-R-B-S-C, llegamos a obtener los mismos valores de Z que si vamos de A a C por T. Pero, por las cardinalidades dadas, si entramos por B y tomamos un valor Y, y recorremos el camino B-S-C-T-A, llegamos a obtener los mismos valores de X que si vamos de B a A directamente por R. Por el contrario, si entramos por C, y recorremos el camino C-T-A-R-B, asociamos a cada Z unos valores de Y que pueden no coincidir con los que se le asocian yendo de C a B directamente por S. Por tanto, R es redundante con S y T, pero S no lo es con R y T.

Por consiguiente, S no puede ser eliminado del diagrama. En cambio, puesto que R y T son redundantes, podemos eliminar una cualquiera de ellas, en principio. Veamos qué diseño se obtiene en ambos casos.

Si eliminamos R, el diagrama queda:



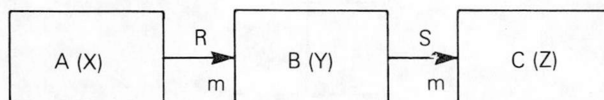
Se transforma en:



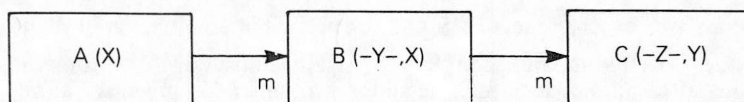
Suprimiendo rectángulos se llega a: C(–Z–, X, Y). Por otra parte, por las cardinalidades dadas debe cumplirse en la relación C la dependencia Y→X, por lo que C no está normalizada.

Para normalizarla, la proyectamos sobre Y, X, y sobre Y, Z. Sean B(Y, X) y C(Y, Z) estas proyecciones. El diseño estará en definitiva formado por las relaciones: B(–Y–, X), C(–Z–, Y).

Veamos ahora qué diseño se obtiene si eliminamos T en el diagrama inicial. Este queda así:



Se transforma en:



Se llega directamente al mismo diseño normalizado que antes.

METODO Y EJEMPLOS

En los ejercicios siguientes, para simplificación de los dibujos, dejaremos a veces las asociaciones explícitas sin nombre. En este caso indicaremos que son explícitas adjuntándoles el signo &. También, por la misma razón, se omitirán a veces los nombres de los conjuntos de valores representados en los rectángulos.

1) *Sean los conjuntos de valores siguientes:*

C1 (NE): Nombres de los empleados de una empresa.

C2 (#D): Números identificadores de los departamentos de la empresa.

C3 (ND): Nombres de los departamentos de la empresa.

C4 (#T): Números de teléfono, dentro de la empresa, asignados a empleados.

C5 (CJ): Código de si un empleado es jefe o no. Es un conjunto de dos valores: S (si es jefe), N (si no es jefe).

Todos los empleados tienen nombres diferentes.

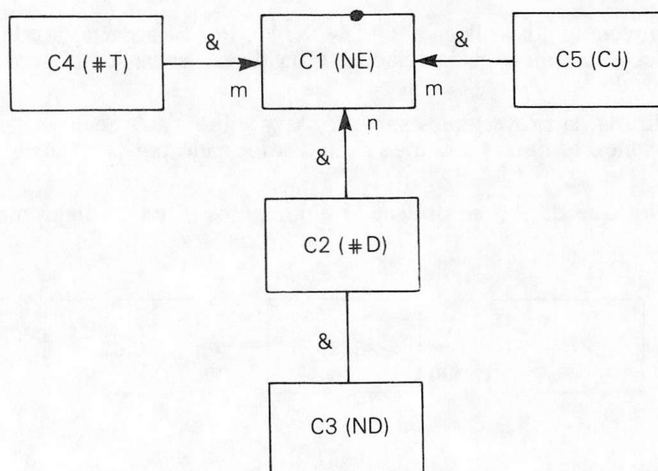
Un departamento puede tener varios empleados o ninguno. Un empleado tiene que estar en un departamento y no puede pertenecer a más de uno.

Un departamento no puede tener más de un jefe, aunque puede estar sin jefe en algún momento.

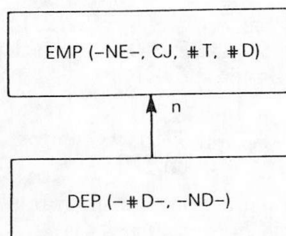
Cada empleado tiene un teléfono. Si es jefe, lo usa en exclusiva, si no, comparte su uso con otros empleados no jefes.

Representar la descripción anterior en un diagrama y obtener el esquema de diseño.

Empecemos dibujando el diagrama:



Se transforma en el siguiente:



El esquema de diseño obtenido es:

- Entidades: Empleados (EMP) y Departamentos (DEP). DEP es independiente, EMP depende de DEP.
- Relaciones: Empleados EMP (-NE-, CJ, #T, #D) y Departamentos: DEP (-#D-, -ND-). En Esta última designamos #D como clave primaria.
- Condiciones adicionales deducibles del diagrama: todo valor de #D de EMP debe existir en DEP.

Condiciones adicionales del enunciado no deducibles del diagrama ni utilizadas en su construcción:

a) Que los jefes no comparten sus teléfonos.

b) Que los departamentos no puedan tener más de un jefe.

Estas condiciones, aplicadas al diseño obtenido, pueden enunciarse así:

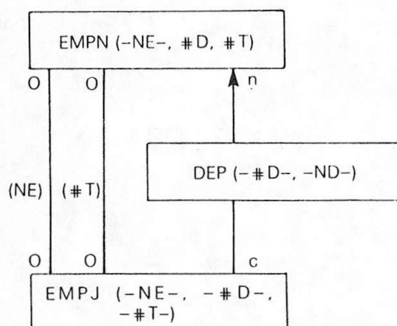
a) Si seleccionamos en EMP todas las filas de los jefes, el atributo #T no se repite, es decir, es una clave.

b) Si seleccionamos en EMP todas las filas de los jefes, el atributo #D no se repite, es decir, es una clave.

Para explicitar estas condiciones en el esquema de diseño hay que partir EMP en dos relaciones, una con los empleados jefes y otra con los no jefes. Las llamaremos EMPJ y EMPN.

En ellas el atributo CJ es redundante, pues siempre toma el mismo valor. Por tanto, lo suprimimos.

El diagrama final quedará entonces así:



F

El esquema de diseño obtenido es:

- Entidades: Empleados que son jefes (EMPJ), Empleados que no son jefes (EMPN) y Departamentos (DEP). DEP es independiente, EMPN depende de DEP y EMPJ es subordinada a DEP.
- Relaciones: Empleados no jefes:EMPN (–NE–, #T, #D). Empleados jefes: EMPJ (–NE–, –#T–, –#D–). Departamentos: DEP (–#D–, –ND–). Claves primarias: NE en EMPJ y #D en DEP.
- Condiciones adicionales deducibles del diagrama:

  Todos los valores de #D de EMPJ y EMPN deben existir en DEP.

  Los valores de #T de EMPJ no pueden existir en EMPN.

  Los valores de NE de EMPJ no pueden existir en EMPN.

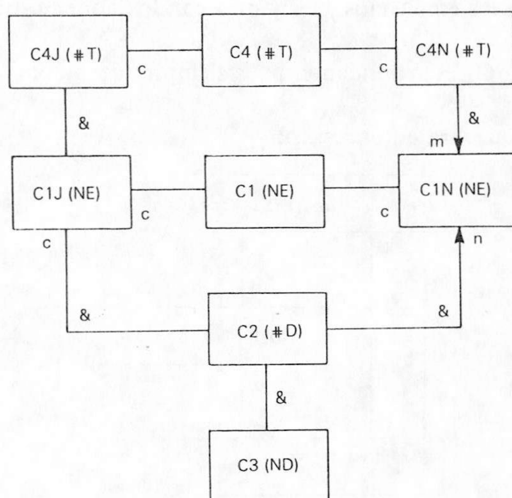
Veamos otra forma de enfocar el problema. Al dibujar el diagrama inicial anterior pusimos cardinalidad (1 : m) a la asociación entre C1 y C4, lo cual no refleja fielmente las condiciones dadas, pues para los jefes esta cardinalidad es (1 : 1). La causa de esta deficiencia de representación es que dentro del conjunto de empleados hay dos subtipos, los que son jefes y los que no, con distintas características, que deseamos reflejar en nuestro modelo de datos. Para conseguirlo hay que dibujar separadamente a cada subtipo o categoría.

Puesto que los teléfonos se asocian a los jefes con una cardinalidad distinta que a los empleados, también los clasificaremos en dos subtipos: los teléfonos asignados a jefes y el resto.

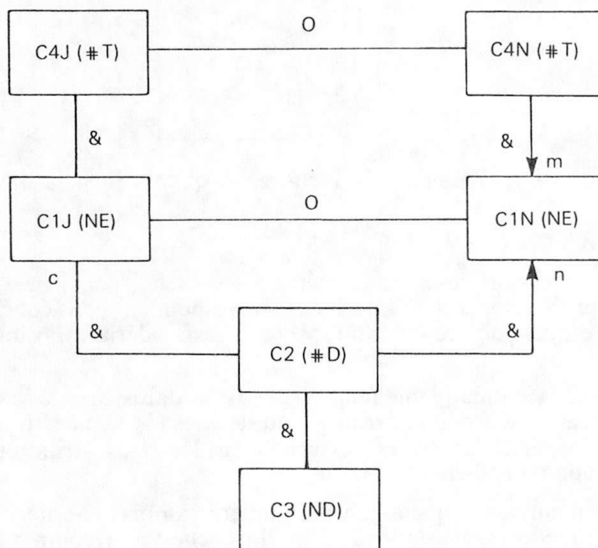
De este modo tendremos los siguientes conjuntos de valores:

- C1J (NE): Nombres de los empleados jefes.
- C1N (NE): Nombres de los empleados que no son jefes.
- C4J (#T): Teléfonos de jefes.
- C4N (#T): Teléfonos de los que no son jefes.

El diagrama inicial quedará así:



En este diagrama, la cardinalidad (1 : c) de la asociación implícita entre C1 y C1J indica que C1J es un subconjunto de C1. Lo mismo es aplicable a C1N. Por otra parte, C1J y C1N incluyen todos los valores de C1, por lo que éste es redundante y lo podemos suprimir del dibujo. Además, no hay valores comunes entre C1J y C1N. Esto lo expresaremos poniendo una asociación entre ellos de cardinalidad 0 (cero). Esto mismo es aplicable a C4, C4J y C4N. El diagrama quedará, por tanto, así:



Aplicando a este diagrama las reglas de transformación obtenemos el mismo esquema de diseño que anteriormente.

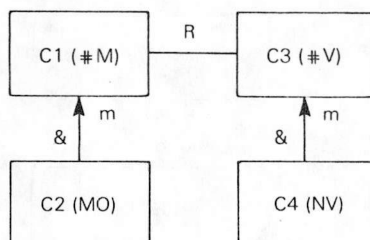
2) *Sean los conjuntos de valores siguientes:*

C1 (#M): Matrículas de los coches de una empresa asignados a sus vendedores.
C2 (MO): Modelos de estos coches.
C3 (#V): Identificadores de vendedores.
C4 (NV): Nombres de vendedores.

La empresa dispone de una flota de coches para sus vendedores. A cada vendedor se le asigna un coche, y cada coche sólo se asigna a un vendedor. Sea R (#M, #V) la asociación entre vendedores y coches.

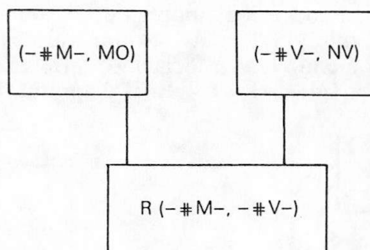
Representar esta descripción en un diagrama y obtener el esquema de diseño.

Diagrama de partida:



F

Se transforma en:



Agregando se transforma en un solo rectángulo, que da lugar a un esquema con una sola relación

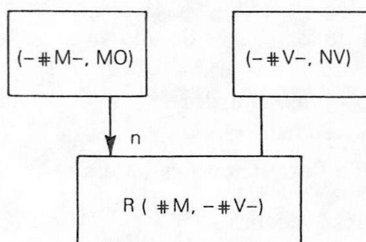
Vendedores: VEN (-#V-, NV, -#M-, MO).

Como ya se ha dicho al describir el método de diseño, conviene validar el esquema obtenido comprobando si las Entidades corresponden con conceptos lógicos del mundo real que queremos representar, y cómo respondería a posibles cambios en las cardinalidades.

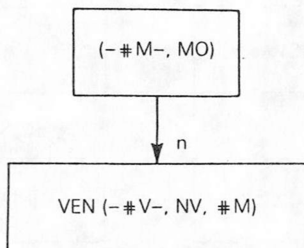
En nuestro caso, la entidad obtenida, VEN, tiene datos mezclados de vendedores y de coches. Esto se debe a que la cardinalidad de R es (1 : 1). Si esto fuera a mantenerse así rigurosamente en el futuro, es correcto considerar los atributos de coches como parte de la Entidad Vendedor.

Supongamos, sin embargo, que algún día pudiera cambiar el criterio de asignación de coches a los vendedores, permitiendo que un coche fuera compartido por varios de ellos. Además, algunos coches pueden estar sin asignar ningún vendedor durante un período de tiempo.

¿Cómo modificaría esto al diseño obtenido? Tendríamos el diagrama de partida:



Se transforma en:

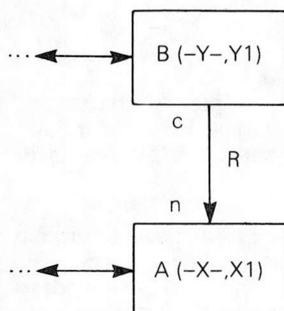


De este modo obtenemos el esquema:

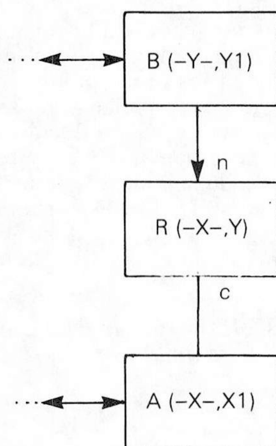
- Entidades: Coches (CCH) y Vendedores (VEN). CCH es independiente, VEN depende de CCH.
- Relaciones: Vendedores: VEN (–#V–, NV, #M). Coches: CCH (–#M–, MO).
- Condiciones deducibles del diagrama: Todos los valores de #M de VEN deben existir en CCH.

3) La Regla de Propagación de Claves, tal como se enunció en el capítulo correspondiente, se aplica entre dos rectángulos ligados por una asociación explícita de cardinalidad (1 : n). *Demostrar que también puede aplicarse cuando la cardinalidad es (c : n) si permitimos que el atributo propagado tome valores nulos.*

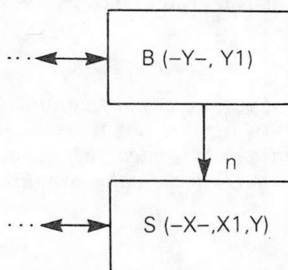
Sea el diagrama siguiente:



Transformemos la asociación explícita R en implícita:



Apliquemos ahora la Regla de Agregación de rectángulos generalizada para valores nulos. Si S es la yunción externa entre A y R, el diagrama queda:

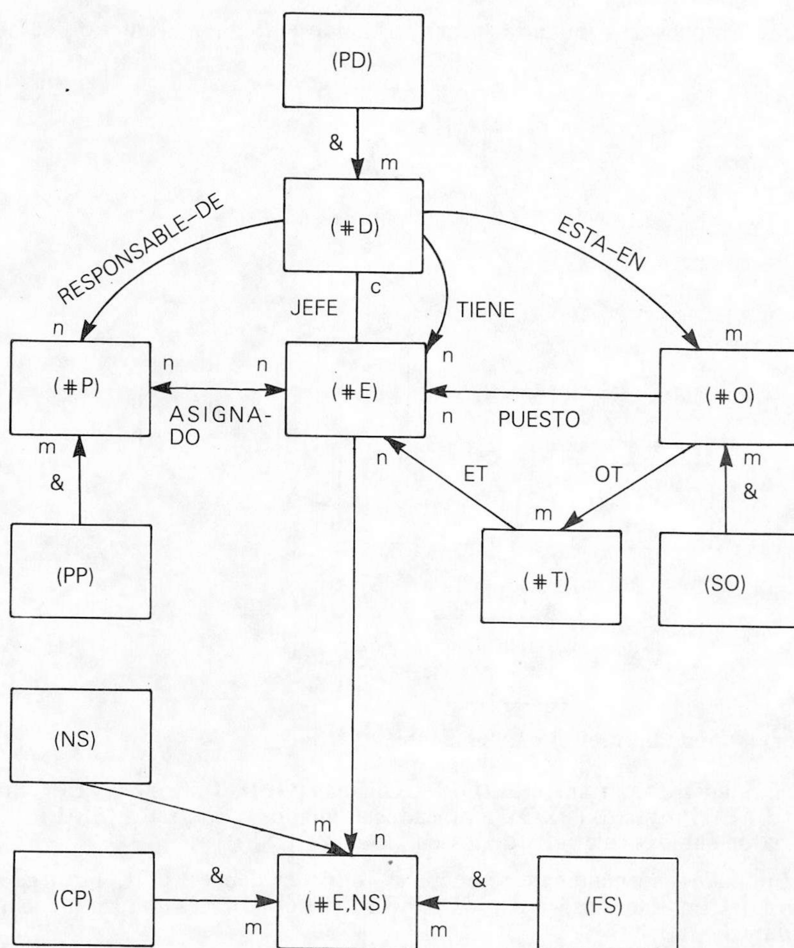


La clave Y se ha propagado a S, pero puede tener valores nulos. Obsérvese además que la cardinalidad original, (c : n), se ha transformado en (1 : n), significando que todos los valores de Y de S que no son nulos existen en B.

4) Queremos diseñar una Base de Datos para almacenar información sobre los Departamentos, Empleados y Oficinas de una empresa. Sus identificadores respectivos son #D, #E y #O. Por cada Departamento se desea almacenar su presupuesto (lo designaremos por PD) en pesetas, y el número identificador del empleado que lo dirige. También se desea guardar por Departamento los identificadores de sus empleados, los de las oficinas que ocupa y los Proyectos de los que cada Departamento es responsable. Los proyectos tienen un identificador llamado #P. Para cada empleado se deseasaber los proyectos a los que está asignado, la oficina donde está su puesto de trabajo, y por tanto donde recibe su correspondencia, y su número de teléfono en dicha oficina. Cada proyecto tiene un presupuesto, llamado PP, en pesetas, no incouyendo en el del Departamento responsable del proyecto. Cada oficina tiene una superficie en metros cuadrados que se desea almacenar también, con el nombre SO. Para cada empleado se desea guardar una información histórica de los aumentos salariales que ha tenido, independientemente de si han ido acompañados de ascenso o no. Por cada subida de sueldo se guardará la fecha en que se ha producido (la llamaremos FS), la categoría profesional del empleado en ese momento (CP) y el nuevo sueldo en pesetas (NS). También se desea saber para cada oficina sus números de teléfono (los números de teléfono los designaremos como #T). Un Departamento puede tener varios empleados, proyectos y oficinas, pero a cada uno de éstos sólo corresponde un Departamento. Un teléfono está en una oficina. Una oficina puede tener varios teléfonos.

Obtener el esquema relacional de diseño de la Base de Datos de acuerdo con esta descripción, completándola si hace falta con las hipótesis adicionales que se crean razonables.

Partimos del siguiente diagrama:



El significado de los nombres de las relaciones explícitas es el siguiente:

RESPONSABLE-DE: Asocia cada Departamento con los Proyectos de que es responsable.

JEFE: Asocia cada Departamento con el empleado que lo dirige.

TIENE: Asocia cada Departamento con los empleados que le pertenecen administrativamente.

ESTA-EN: Asocia cada Departamento con las oficinas en que está instalado.

ASIGNADO: Asocia cada empleado con los proyectos en los que trabaja.

PUESTO: Asocia cada empleado con la oficina donde se encuentra instalado.

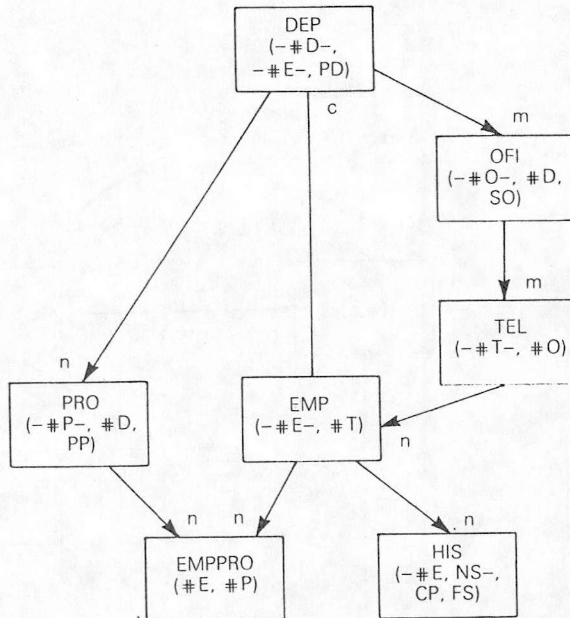
OT: Asocia a cada oficina con sus teléfonos.

ET: Asocia a cada empleado con su teléfono.

Parece razonable suponer que la asociación TIENE es redundante con ESTA-EN y PUESTO. La podemos eliminar del dibujo. A su vez, PUESTO es redundante con

F

OT y ET. Después de eliminarla y transformando el diagrama anterior, se obtiene el siguiente:



Obtenemos, por tanto, el diseño siguiente:

a) Entidades: Departamentos (DEP), Oficinas (OFI), Teléfonos (TEL), Empleados (EMP), Proyectos (PRO), Empleados asignados a Proyectos (EMPPRO),Historia de subidas salariales de los empleados (HIS).

Entidades independientes: Ninguna. Entidades débiles: OFI, TEL, EMP, PRO y HIS. Entidades asociativas: DEP, EMPPRO. DEP es una Entidad subordinada a EMP.

b) Relaciones:

DEP (–#D–, –#E–, PD), (#E: identificador del jefe del Departamento);
OFI (–#O–, #D, SO);
TEL (–#T–, #O);
EMP (–#E–, #T);
PRO (–#P–, #D, PP);
HIS (–#E, NS–, CP, FS);
EMPPRO (#E, #P).
En DEP designamos a #D como clave primaria (más estable que #E).

c) Condiciones adicionales expresadas en el diagrama:

Los valores de #D en OFI deben existir en DEP, y viceversa.
Los valores de #O en TEL deben existir en OFI, y viceversa.
Los valores de #T en EMP deben existir en TEL.
Los valores de #E en DEP deben existir en EMP.
Los valores de #D en PRO deben existir en DEP.
Los valores de #E en EMPPRO deben existir en EMP.

F

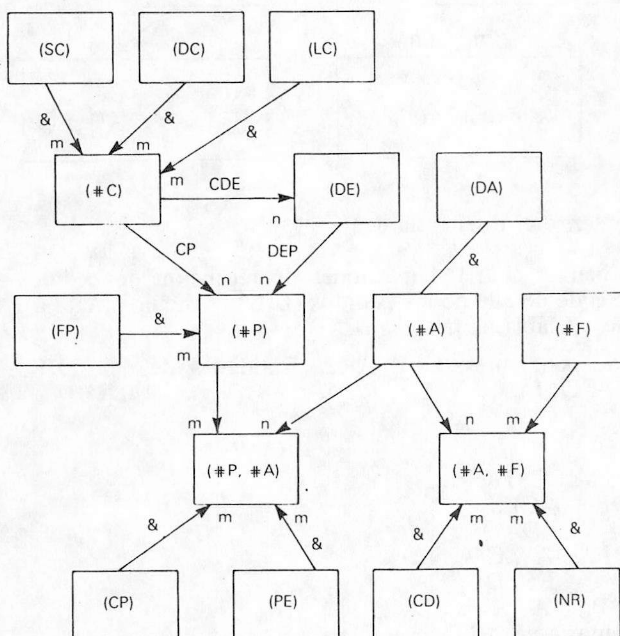
Los valores de #P en EMPPRO deben existir en PRO.
Los valores de #E en HIS deben existir en EMP.

d) Condición no expresada en el diagrama: (#E, FS) es una clave en HIS.

5) Queremos diseñar una Base de Datos para almacenar información sobre nuestros Clientes y sus Pedidos, cuyos identificadores respectivos son #C y #P. Los artículos comercializados por nuestra organización tienen un identificador llamado #A. Algunos son fabricados en factorías propias, cuyo identificador es #F, y otros hay que adquirirlos en proveedores externos. Cada cliente dispone de un crédito cuyo límite se llama LC, de un descuento llamado DC por compras masivas, y de un saldo, SC, cuyo importe varía conforme va pagando los pedidos entregados. Los pedidos se entregan en una dirección de entre varias posibles de cada cliente. Estas direcciones de llaman DE (dirección de entrega). Los pedidos se componen de dos partes: una cabecera de pedido y varias líneas de detalle. En la cabecera figuran la fecha del pedido que lo hace. Las líneas de detalle contienen el artículo pedido y la cantidad pedida. Se desea guardar toda esta información en la Base de Datos. Cada artículo tiene una descripción única. Se desea también llevar un control de las necesidades de fabricación. Para ello, por cada fábrica propia se guardará el nivel de inventario en el almacén de la fábrica de cada artículo (CD: Cantidad Disponible) y el nivel mínimo de existencias a partir del cual hay riesgo de quedarse sin existencias (NR: Nivel de Riesgo). Un pedido puede servirse en varias entregas. Para llevar un control de éstas, se guardará por cada línea de detalle un contador, con lo que queda pendiente de entregar en cada momento, que se llama PE.

Obtener el esquema relacional de diseño de la Base de Datos de acuerdo con esta descripción, completándola si hace falta con las hipótesis adicionales que se crean razonables.

Partimos del siguiente diagrama:



F

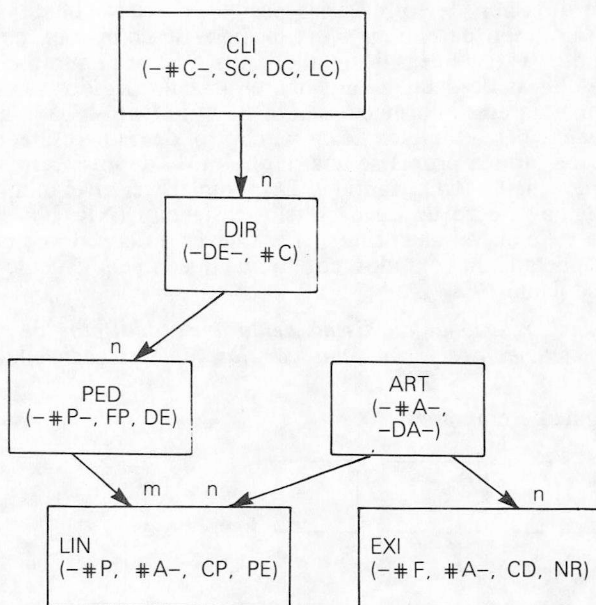
El significado de los nombres de las relaciones explícitas es el siguiente:

CDE: Asociación (Cliente-Dirección de envío). Asocia a cada cliente con todas sus posibles direcciones de recepción de pedidos.

DEP: Asociación (Direción de envío-Pedido). Asocia a cada pedido con la dirección donde hay que entregarlo.

CP: Asociación (Cliente-Pedido). Asocia a cada cliente con sus pedidos.

Es lógico suponer que la asociación CP es redundante con CDE y DEP. La eliminamos del dibujo. Aplicando a éste las reglas de transformación se obtiene el siguiente:



Obtenemos por tanto el diseño siguiente:

a) Entidades: Clientes (CLI), Direcciones de recepción de pedidos (DIR), Pedidos (PED), Líneas de detalle de los pedidos (LIN), Artículos (ART), Existencias en los almacenes de las fábricas propias (EXI).

Entidades independientes: CLI, ART. Entidades débiles: DIR, EXI. Entidades asociativas: PED, LIN.

b) Relaciones:

CLI (–#C–, SC, DC, LC);
DIR (–DE–, #C);
ART (–#A–, –DA–);
EXI (–#F, #A–, CD, NR);
PED (–#P–, FP, DE);
LIN (–#P, #A–, CP, PE).
Clave primaria en ART: #A.

F

c) Condiciones adicionales expresadas en el diagrama:

Los valores de #C en DIR deben existir en CLI.
Los valores de DE en PED deben existir en DIR.
Los valores de #P en LIN deben existir en PED, y viceversa.
Los valores de #A en LIN deben existir en ART.
Los valores de #A en EXI deben existir en ART.

d) Condiciones adicionales no expresadas en el diagrama:

Los valores de PE no deben superar a los de CP.
Los valores de LC no deben superar a 0, ni a los de SC.
Los valores de CP, PE, NR, DC y CD no pueden ser negativos.

Otros libros de INFORMATICA publicados por



Obras generales

ABRAMSON.– Teoría de la información y codificación.
ANGULO y ZAPATER.– Introducción a la informática.
BANKS.– Microordenadores. Cómo funcionan. Para qué sirven.
COHEN.– Estructura lógica y diseño de programas.
FLORES.– Estructuración y proceso de datos.
GALAN-CORDERO.– Teleinformática.
GARCIA SANTESMASES.– Cibernética. Aspectos y tendencias actuales.
GARDARIN.– Bases de datos.
GOSLING.– Códigos para ordenadores y microprocesadores.
HARTMAN.– Manual de los sistemas de información. 2 tomos.
HUNT.– Manual de informática básica.
HURTADO MERELO.– 200 problemas de informática.
INTERTAGLIA.– Guía práctica del comando numérico.
LEWIS.– Estructuras de datos. Programación y aplicaciones.
LUCAS.– Sistemas de información. Análisis. Diseño. Puesta a punto.
MENASCE.– Redes de computadores.
PEREZ-LEMAUR.– Diagramas de flujo. Ejercicios y problemas.
POPKIN.– Introducción al proceso de datos.
PUJOLLE.– Telemática.
RODRIGUEZ.– Protección de la información. Diseño de criptosistemas informáticos.
SCHMIDT.– Introducción a los ordenadores y al proceso de datos.

Diccionarios

HART.– Diccionario del Basic.
NANIA.– Diccionario de Informática. Inglés, francés, español. Más de 11.000 términos. Cartoné
OLIVETTI.– Diccionario de Informática. Inglés-Español.

Lenguajes

ALONSO.– Técnicas de programación. Programación estructurada. Pascal. Estructuras de datos.
BELLIDO y SANCHEZ.– Basic para estudiantes.
BELLIDO y SANCHEZ.– Basic para Maestros.
BELLIDO.– Basic para maestros (Libro y casete)
BURKE.– Enseñanza asistida por ordenador.
CECIL.– Depuración de programas en BASIC.
CUÑAT.– Pascal para estudiantes.
CUÑAT.– Pascal y Turbopascal. Programas y prácticas.
CHECROUN.– Basic. Programación de microordenadores.
DAX.– Lenguaje C.
DELANNOY.– Ficheros en Basic.
GALAN PASCUAL.– MODULA 2. Desarrollo de Software. (Con disquete).
GALAN PASCUAL.– Pascal y turbopascal.
GALAN PASCUAL.– Programación con el lenguaje Cobol.
GARCIA MERAYO.– Programación en Fortran 77.
GOMEZ, DEL PINO.– Curso completo de BASIC. T.I. Fundamentos.
GOMEZ, DEL PINO.– Curso completo de BASIC. T. II. Métodos.
GOMEZ, DEL PINO.– Curso completo de BASIC. T.III. Ficheros.

HURTADO MERELO.– Cobol con programación estructurada. Curso acelerado.
HURTADO MERELO.– Lenguaje Cobol con programación estructurada.
HURTADO MERELO.– Programas Cobol. Incluye ejercicios de programación estructurada.
LARRECHE.– Basic. Introducción a la programación.
LEPAPE.– Programación del Z80 con ensamblador.
MARSHALL.– Lenguajes de programación para Micros.
Mc NITT.– Simulación con ordenador.
MONTEIL.– Primeros pasos en Logo.
MORALES.– Problemas resueltos de programación estructurada.
NUÑEZ, GARCIA, COLLADO, HERNANDEZ.– Logo. Aprender a pensar.
OAKEY.– Lenguajes Forth para micros.
QUANEAUX.– Tratamiento de textos con Basic.
QUEINNEC.– Programación en Lisp.
RAMON.– 44 Superprogramas en Basic.
ROSSI.– Basic. Curso acelerado.
SANCHIS y MORALES.– Programación con el lenguaje Pascal.
SARAM, DE.– Programación en micro-prolog. Un lenguaje de la 5ª generación.
WATT y MANGADA.– Basic avanzado para niños.
WATT y MANGADA.– Basic para niños.
WATT, MANGADA y GOMEZ-MASCARAQUE.– Logo para niños.
YOUNG.– Lenguajes en tiempo real.

Aplicaciones profesionales

ANGELL.– Gráficos con computador.
ANGULO/NO.– Control de procesos industriales por computador.
ASTROM y WITTENMA.– Sistemas controlados por computador.
BARRIO.– Subrutinas para aumentar la precisión de cálculo.
BUITELAAR.– Informática y medicina.
CROCUS.– Sistemas de explotación de computadores.
GAUTHIER.– Diseño de programas para sistemas.
GRIGORIEFF.– DBASE II DBASE III. Guía de uso.
GRIGORIEFF.– Informática para actividades profesionales.
LANE.– Telemática y comunicaciones en la empresa.
NOWAKOWSKI.– Electro BASIC.
O'REILLY.– Ingeniería electrónica asistida por computador.
PANNELL.– El microordenador en la pequeña empresa.
THOMAS y DOUGLAS.– Auditoria informática.
THORIN.– Ingeniería del Software.
WHITAKER.– Investigación operativa con el computador.

Colección 99%

ALONSO.– dBASE III Plus.
ALONSO.– Lotus 1-2-3.
GOMEZ-MASCARAQUE, GOMEZ.– Open Access II.
IBAÑEZ.– Wordperfect.
KAMIN.– Disco duro.
LOPEZ.– Multiplán.
LOPEZ-BAISSON.– Serie Assistant de IBM.
LOPEZ-BAISSON.– Symphony.
LOPEZ, GOMEZ-MASCARAQUE.– Wordstar.
NAVAS.– Autocad.
TORRES.– Clipper.

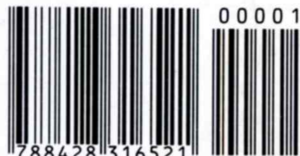
BASES DE DATOS RELACIONALES

A lo largo de los últimos años se está expandiendo rápidamente la utilización de bases de datos relacionales en toda la escala de sistemas informáticos, desde los ordenadores personales a los grandes sistemas. Hoy día prácticamente todas las instalaciones informáticas necesitan profesionales formados en esta técnica.

El primer paso, y de fundamental importancia para utilizar bien una base de datos relacional es hacer un buen diseño de sus estructuras de datos. A ello va encaminado el presente libro. En él se presentan los conceptos básicos del Modelo Relacional de Datos, la teoría de Normalización y un método de diseño basado en gráficos de Entidades, Relacionales y Atributos.

Se ha buscado un estilo didáctico en la presentación de estos temas. Para ello, los nuevos conceptos y definiciones se introducen justificándolos en función del problema que pretenden resolver. Se ha incluido además un gran número de ejemplos y ejercicios, con sus soluciones. Todo ello hace de este libro un valioso instrumento de aprendizaje y entrenamiento tanto para los estudiantes (Facultades de Informática, Escuelas Técnicas, Escuelas Universitarias, etc.) como para los profesionales informáticos que necesiten utilizar bases de datos relacionales.

ISBN 84-283-1652-X





Magallanes, 25 - 28015 Madrid

BASES DE DATOS RELACIONALES E. Rivero Cornelio

PARANINFO S.A.